

ENCYCLOPEDIA OF  
DATABASE  
TECHNOLOGIES  
AND APPLICATIONS



Laura Rivero, Jorge Doorn & Viviana Ferraggino

TEAMLING

# Encyclopedia of Database Technologies and Applications

Laura C. Rivero

*Universidad Nacional del Centro de la Provincia de  
Buenos Aires, Argentina*

Jorge H. Doorn

*Universidad Nacional del Centro de la Provincia de  
Buenos Aires, Argentina*

Viviana E. Ferraggine

*Universidad Nacional del Centro de la Provincia de  
Buenos Aires, Argentina*



**IDEA GROUP REFERENCE**  
Hershey • London • Melbourne • Singapore

Acquisitions Editor: Renée Davies  
Development Editor: Kristin Roth  
Senior Managing Editor: Amanda Appicello  
Managing Editor: Jennifer Neidig  
Copy Editors: Beth Arneson, April Schmidt and Becky Shore  
Typesetters: Diane Huskinson, Sara Reed and Larissa Zearfoss  
Support Staff: Michelle Potter  
Cover Design: Lisa Tosheff  
Printed at: Yurchak Printing Inc.

Published in the United States of America by  
Idea Group Reference (an imprint of Idea Group Inc.)  
701 E. Chocolate Avenue, Suite 200  
Hershey PA 17033  
Tel: 717-533-8845  
Fax: 717-533-8661  
E-mail: [cust@idea-group.com](mailto:cust@idea-group.com)  
Web site: <http://www.idea-group-ref.com>

and in the United Kingdom by  
Idea Group Reference (an imprint of Idea Group Inc.)  
3 Henrietta Street  
Covent Garden  
London WC2E 8LU  
Tel: 44 20 7240 0856  
Fax: 44 20 7379 3313  
Web site: <http://www.eurospan.co.uk>

Copyright © 2006 by Idea Group Inc. All rights reserved. No part of this publication may be reproduced, stored or distributed in any form or by any means, electronic or mechanical, including photocopying, without written permission from the publisher.

Product or company names used in this set are for identification purposes only. Inclusion of the names of the products or companies does not indicate a claim of ownership by IGI of the trademark or registered trademark.

#### Library of Congress Cataloging-in-Publication Data

Encyclopedia of database technologies and applications / Laura Rivero, Jorge Doorn and Viviana Ferraggine, editors.  
p. cm.

Summary: "This encyclopedia will be a collection of diverse topics related to databases, providing an overview not only of the state of the art of classical subjects but also clear and concise explanations of the latest technology, its practical applications and research directions"--Provided by publisher.

Includes bibliographical references and index.

ISBN 1-59140-560-2 (hardcover) -- ISBN 1-59140-795-8 (ebook)

1. Database management--Encyclopedias. I. Rivero, Laura, 1956- II. Doorn, Jorge H., 1946- III. Ferraggine, Viviana, 1961- QA76.9.D3E624 2005  
005.74'03--dc22

2005004545

#### British Cataloguing in Publication Data

A Cataloguing in Publication record for this book is available from the British Library.

All work contributed to this encyclopedia set is new, previously-unpublished material. The views expressed in this encyclopedia set are those of the authors, but not necessarily of the publisher.

# Editorial Advisory Board

Robert Laurini, *LIRIS, France*

Christine Parent, *Universite de Lausanne, Switzerland*

Makoto Takizawa, *Tokyo Denki University, Japan*

Aziz Barbar, *American University College of Science & Technology, Lebanon*

Jaroslav Pokorny, *Charles University in Prague, Czech Republic*

M. Tamer Özsu, *University of Waterloo, Canada*

Alejandro Buchmann, *Technische Universität Darmstadt, Germany*

Alexander Osterwalder, *Institut d'Informatique et Organisation (INFORGE), Switzerland*



# List of Contributors

**Abidin, Taufik** / *North Dakota State University, USA*  
**Aldana-Montes, José F.** / *University of Malaga, Spain*  
**Ale, Juan M.** / *Universidad de Buenos Aires, Argentina*  
**Aleman-Meza, Boanerges** / *University of Georgia, USA*  
**Alhajj, Reda** / *University of Calgary, Canada*  
**Alonso-Jiménez, José A.** / *Universidad de Sevilla, Spain*  
**Arpinar, I. Budak** / *University of Georgia, USA*  
**Artz, John M.** / *The George Washington University, USA*  
**Asprey, Len** / *Practical Information Management Solutions Pty Ltd., Australia*  
**Badia, Antonio** / *University of Louisville, USA*  
**Bagui, Sikha** / *University of West Florida, USA*  
**Balsters, H.** / *University of Groningen, The Netherlands*  
**Banaei-Kashani, Farnoush** / *University of Southern California, USA*  
**Baumann, Peter** / *International University Bremen, Germany*  
**Berberidis, Christos** / *Aristotle University of Thessaloniki, Greece*  
**Borges, Karla A. V.** / *Empresa de Informática e Informação do Município de Belo Horizonte and Universidade Federal de Minas Gerais, Brazil*  
**Borrego-Díaz, Joaquín** / *Universidad de Sevilla, Spain*  
**Bose, Indranil** / *University of Hong Kong, Hong Kong*  
**Bounif, Hassina** / *Swiss Federal Institute of Technology, Switzerland*  
**Buccella, Agustina** / *Universidad Nacional del Comahue, Argentina*  
**Buchmann, Alejandro** / *Technische Universität Darmstadt, Germany*  
**Bussler, Christoph** / *Digital Enterprise Research Institute (DERI), Ireland*  
**Camolesi, Jr., Luiz** / *Methodist University of Piracicaba, Brazil*  
**Cannataro, Mario** / *Magna Graecia University of Catanzaro, Italy*  
**Cartelli, Antonio** / *University of Cassino, Italy*  
**Cechich, Alejandra** / *Universidad Nacional del Comahue, Argentina*  
**Chang, Shi Kuo** / *University of Pittsburgh, USA*  
**Chávez, Edgar** / *Universidad Michoacana, Mexico*  
**Chávez-González, Antonia M.** / *Universidad de Sevilla, Spain*  
**Chen, Yangjun** / *University of Winnipeg, Canada*  
**Chen, Zhengxin** / *University of Nebraska at Omaha, USA*  
**Cheng, Reynold** / *Purdue University, USA*  
**Chengalur-Smith, InduShobha** / *University at Albany, USA*  
**Cilia, Mariano A.** / *Technische Universität Darmstadt, Germany and UNICEN, Argentina*  
**Cluet, Sophie** / *INRIA Rocquencourt-Xyleme S.A., France*  
**Corral, Antonio** / *University of Almeria, Spain*  
**Costagliola, Gennaro** / *Università di Salerno, Italy*  
**Curti, Hugo J.** / *Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina*  
**Dadashzadeh, Mohammad** / *Oakland University, USA*  
**Davis, Jr., Clodoveu A.** / *Pontificia Universidade Católica de Minas Gerais and Empresa de Informática e Informação do Município de Belo Horizonte, Brazil*

**Decker, Hendrik** / *Instituto Tecnológico de Informática, Spain*  
**Deshpande, Prasad M.** / *IBM Almaden Research Center, USA*  
**Deufemia, Vincenzo** / *Università di Salerno, Italy*  
**Dholakia, Nikhilesh** / *University of Rhode Island, USA*  
**Ding, Qiang** / *Concordia College, USA*  
**Diskin, Zinovy** / *SWD Factory, Latvia and Universal Information Technology Consulting, USA*  
**Dunn, Cheryl L.** / *Grand Valley State University, USA*  
**Englebert, Vincent** / *University of Namur, Belgium*  
**Faiz, Sami** / *National Institute of Applied Sciences and Technology, Tunisia*  
**Fernandez Ausinaga, José Luis** / *Pop-Vision Company, Argentina*  
**Ferri, Fernando** / *Istituto di Ricerche sulla Popolazione e le Politiche Sociali, Italy*  
**Fiege, Ludger** / *Technische Universität Darmstadt, Germany*  
**Flesca, Sergio** / *DEIS Università della Calabria, Italy*  
**Frank, Lars** / *Copenhagen Business School, Denmark*  
**Fulton, James A.** / *The Boeing Company, USA (Retired)*  
**Furfaro, Fillippo** / *DEIS Università della Calabria, Italy*  
**Gabillon, Alban** / *IUT de Mont de Marsan, France*  
**Gaffar, Ashraf** / *Concordia University, Canada*  
**Galitsky, Boris** / *Birkbeck College University of London, UK*  
**González Císaro, Sandra Elizabeth** / *Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina*  
**Grabski, Severin V.** / *Michigan State University, USA*  
**Greco, Sergio** / *DEIS Università della Calabria, Italy*  
**Green, Rolf** / *OneView Pty Ltd., Australia*  
**Gupta, P.** / *Indian Institute of Technology, India*  
**Hainaut, Jean-Luc** / *University of Namur, Belgium*  
**Halaschek-Wiener, Christian** / *University of Georgia, USA*  
**Haraty, Ramzi A.** / *Lebanese American University, Lebanon*  
**Henrard, Jean** / *REVER S.A., Belgium*  
**Hentea, Mariana** / *Southwestern Oklahoma State University, USA*  
**Hernandez-Orallo, Jose** / *Technical University of Valencia, Spain*  
**Hick, Jean-Marc** / *REVER S.A., Belgium*  
**Horiuchi, Catherine** / *Seattle University, USA*  
**Ibrahim, Hamidah** / *Universiti Putra Malaysia, Malaysia*  
**Irún-Briz, Luis** / *Instituto Tecnológico de Informática, Spain*  
**Kadish, Boris** / *SWD Factory, India*  
**Kaushik, A.K.** / *Ministry of Communication and Information Technology, India*  
**Kersten, Martin** / *Center for Mathematics and Computer Science, The Netherlands*  
**Khan, Sharifullah** / *National University of Sciences and Technology, Pakistan*  
**Kincses, Zoltán** / *Eötvös Loránd University of Sciences, Hungary*  
**Klassen, Chris** / *University of Dallas, USA*  
**Kontaki, Maria** / *Aristotle University of Thessaloniki, Greece*  
**Koutrika, Georgia** / *University of Athens, Greece*  
**Kulikowski, Juliusz L.** / *Institute of Biocybernetics and Biomedical Engineering PAS, Poland*  
**Laender, Alberto H. F.** / *Universidade Federal de Minas Gerais, Brazil*  
**Lee, Choongseok** / *Korea Polytechnic University, Korea*  
**Lee, Heeseok** / *Korea Advanced Institute of Science and Technology, Korea*  
**Leong, Hong Va** / *The Hong Kong Polytechnic University, Hong Kong*  
**Ma, Z.M.** / *Northeastern University, China*  
**Maamar, Zakaria** / *Zayed University, UAE*  
**Mahmoudi, Khaoula** / *High School of Communications-Tunis (SUPCOM), Tunisia*  
**Manegold, Stefan** / *CWI, The Netherlands*  
**Mani, Murali** / *Worcester Polytechnic Institute, USA*  
**Manolopoulos, Yannis** / *Aristotle University of Thessaloniki, Greece*  
**Marjomaa, Esko** / *University of Joensuu, Finland*

**Martínez-González, Mercedes** / *Universidad de Valladolid, Spain*  
**Meixner, Matthias** / *Technische Universität Darmstadt, Germany*  
**Middleton, Michael** / *Queensland University of Technology, Australia*  
**Muñoz-Escóí, Francesc D.** / *Instituto Tecnológico de Informática, Spain*  
**Navarro, Gonzalo** / *University of Chile, Chile*  
**Navas-Delgado, Ismael** / *University of Malaga, Spain*  
**Neely, M. Pamela** / *Rochester Institute of Technology, USA*  
**Nigro, Héctor Oscar** / *Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina*  
**Nørvåg, Kjetil** / *Norwegian University of Science and Technology, Norway*  
**Pandya, Anil** / *Northeastern Illinois University, USA*  
**Papadopoulos, Apostolos N.** / *Aristotle University of Thessaloniki, Greece*  
**Parker, Shin** / *University of Nebraska at Omaha, USA*  
**Pawlak, Zdzislaw** / *Polish Academy of Sciences and Warsaw School of Information Technology, Poland*  
**Perrizo, William** / *North Dakota State University, USA*  
**Ping, Wang** / *University of Hong Kong, Hong Kong*  
**Pinheiro, Francisco A. C.** / *Universidade de Brasília, Brazil*  
**Pires Vieira, Marina Teresa** / *Methodist University of Piracicaba, Brazil*  
**Polat, Faruk** / *Middle East Technical University, Turkey*  
**Polese, Giuseppe** / *Università di Salerno, Italy*  
**Polkowski, Lech** / *Polish-Japanese Institute of Information Technology and University of Warmia and Mazury, Poland*  
**Porta, Gaspar** / *Carthage College, USA*  
**Prabhakar, Sunil** / *Purdue University, USA*  
**R., Manjunath** / *Bangalore University, India*  
**Rafanelli, Maurizio** / *Istituto di Analisi dei Sistemi ed Informatica “A. Ruberti”, Italy*  
**Raisinghani, Mahesh S.** / *Texas Woman’s University, USA*  
**Ramasamy, Karthikeyan** / *Juniper Networks, USA*  
**Rodríguez Brisaboa, Nieves** / *Universidade da Computación, Spain*  
**Rodríguez-Tastets, M. Andrea** / *University of Concepción and University of Chile, Chile*  
**Roland, Didier** / *REVER S.A., Belgium*  
**Roldán-García, María del Mar** / *University of Malaga, Spain*  
**Rossi, Gustavo H.** / *Universidad Nacional de La Plata, Argentina*  
**Roussos, George** / *University of London, UK*  
**Samoladas, Ioannis** / *Aristotle University of Thessaloniki, Greece*  
**Schmidt, Albrecht** / *Aalborg University, Denmark*  
**Schreiber, Zvi** / *Unicorn Solutions Inc., USA*  
**Schwartz, David G.** / *Bar-Ilan University, Israel*  
**Sebastiani, Fabrizio** / *Università di Padova, Italy*  
**Seffah, Ahmed** / *Concordia University, Canada*  
**Sellis, Timos** / *National Technical University of Athens, Greece*  
**Serazi, Masum** / *North Dakota State University, USA*  
**Shahabi, Cyrus** / *University of Southern California, USA*  
**Shan, Mok Wai** / *University of Hong Kong, Hong Kong*  
**Shi, Yong** / *University of Manitoba, Canada*  
**Shing, Wong Ka** / *University of Hong Kong, Hong Kong*  
**Shing, Yip Yee** / *University of Hong Kong, Hong Kong*  
**Simitsis, Alkis** / *National Technical University of Athens, Greece*  
**Singh, Richa** / *Indian Institute of Technology, India*  
**Sirangelo, Cristina** / *DEIS Università della Calabria, Italy*  
**Skowron, Andrzej** / *Warsaw University, Poland*  
**Sowe, Sulayman K.** / *Aristotle University of Thessaloniki, Greece*  
**St.Amant, Kirk** / *Texas Tech University, USA*  
**Stamelos, Ioannis** / *Aristotle University of Thessaloniki, Greece*  
**Stuckenschmidt, Heiner** / *Vrije Universiteit Amsterdam, The Netherlands*  
**Suh, Woojong** / *Inha University, Korea*

**Tin, Chan Lit** / *University of Hong Kong, Hong Kong*  
**Tortora, Genny** / *Università di Salerno, Italy*  
**Tradigo, Giuseppe** / *Magna Graecia University of Catanzaro, Italy*  
**Tribunella, Thomas** / *Rochester Institute of Technology, USA*  
**Trubitsyna, Irina** / *DEIS Università della Calabria, Italy*  
**Tzanis, George** / *Aristotle University of Thessaloniki, Greece*  
**Tzouramanis, Theodoros** / *University of the Aegean, Greece*  
**Uz Tansel, Abdullah** / *Bilkent University, Turkey*  
**Vargas-Solar, Genoveva** / *National Centre for Scientific Research (CNRS), France*  
**Vassilakopoulos, Michael** / *TEI of Thessaloniki, Greece*  
**Vassiliadis, Panos** / *University of Ioannina, Greece*  
**Vatsa, Mayank** / *Indian Institute of Technology, India*  
**Veltri, Pierangelo** / *Magna Graecia University of Catanzaro, Italy*  
**Vlahavas, Ioannis** / *Aristotle University of Thessaloniki, Greece*  
**Vodislav, Dan** / *CNAM Cedric-Lab, Paris, France*  
**Wai, Shiu Ka** / *University of Hong Kong, Hong Kong*  
**Wang, Baoying** / *North Dakota State University, USA*  
**Windhouwer, Menzo** / *Center for Mathematics and Computer Science, The Netherlands*  
**Xodo, Daniel** / *Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina*  
**Zarri, Gian Piero** / *University of Paris IV/Sorbonne, France*  
**Zelasco, José Francisco** / *UNBA, Argentina and UNCPBA, Argentina*  
**Zendulka, Jaroslav** / *Brno University of Technology, Czech Republic*  
**Zhang, Yanpu** / *University of Nebraska at Omaha, USA*  
**Zoumboulakis, Michael** / *University of London, UK*  
**Zumpano, Ester** / *DEIS Università della Calabria, Italy*  
**Zwick, Detlev** / *York University, Canada*

# Contents

Active Database Management Systems / <i>Mariano A. Cilia</i> .....	1
Active Federated Database Systems / <i>Genoveva Vargas-Solar</i> .....	5
Advanced Query Optimization / <i>Antonio Badia</i> .....	11
Applying Database Techniques to the Semantic Web / <i>María del Mar Roldán-García, Ismael Navas-Delgado, and José F. Aldana-Montes</i> .....	18
Benchmarking and Data Generation in Moving Objects Databases / <i>Theodoros Tzouramanis</i> .....	23
Bioinformatics Data Management and Data Mining / <i>Boris Galitsky</i> .....	29
Biological Data Mining / <i>George Tzanis, Christos Berberidis, and Ioannis Vlahavas</i> .....	35
Biometric Databases / <i>Mayank Vatsa, Richa Singh, P. Gupta, and A.K. Kaushik</i> .....	42
Business Rules in Databases / <i>Antonio Badia</i> .....	47
Business-to-Business Integration / <i>Christoph Bussler</i> .....	54
CASE Tools for Database Engineering / <i>Jean-Luc Hainaut, Jean Henrard, Jean-Marc Hick, Didier Roland, and Vincent Englebert</i> .....	59
Checking Integrity Constraints in a Distributed Database / <i>Hamidah Ibrahim</i> .....	66
Collective Knowledge Composition in a P2P Network / <i>Boanerges Aleman-Meza, Christian Halaschek-Wiener, and I. Budak Arpinar</i> .....	74
Common Information Model / <i>James A. Fulton</i> .....	78
Component-Based Generalized Database Index Model / <i>Ashraf Gaffar</i> .....	87
Consistency in Spatial Databases / <i>M. Andrea Rodríguez-Tastets</i> .....	93
Converting a Legacy Database to Object-Oriented Database / <i>Reda Alhajj and Faruk Polat</i> .....	99
Data Dissemination / <i>Ludger Fiege</i> .....	105
Data Model Versioning and Database Evolution / <i>Hassina Bounif</i> .....	110

Data Quality Assessment / <i>Juliusz L. Kulikowski</i> .....	116
Data Warehouses / <i>Antonio Badia</i> .....	121
Data Warehousing and OLAP / <i>Jose Hernandez-Orallo</i> .....	127
Data Warehousing, Multi-Dimensional Data Models, and OLAP / <i>Prasad M. Deshpande and Karthikeyan Ramasamy</i> .....	134
Database Engineering Focusing on Modern Dynamism Crises / <i>Luiz Camolesi, Jr. and Marina Teresa Pires Vieira</i> .....	140
Database Query Personalization / <i>Georgia Koutrika</i> .....	147
Database Replication Protocols / <i>Francesc D. Muñoz-Escoí, Luis Irún-Briz, and Hendrik Decker</i> .....	153
Database Support for Workflow Management Systems / <i>Francisco A. C. Pinheiro</i> .....	158
Databases for Mobile Applications / <i>Indranil Bose, Wang Ping, Mok Wai Shan, Wong Ka Shing, Yip Yee Shing, Chan Lit Tin, and Shiu Ka Wai</i> .....	162
Dataveillance and Panoptic Marketspaces / <i>Nikhilesh Dholakia, Detlev Zwick, and Anil Pandya</i> .....	170
Deriving Spatial Integrity Constraints from Geographic Application Schemas / <i>Clodoveu A. Davis, Jr., Karla A. V. Borges, and Alberto H. F. Laender</i> .....	176
Development Environment for Customer-Oriented Web Business, A / <i>Choongseok Lee, Woojong Suh, and Heeseok Lee</i> .....	184
Digital Media Warehouses / <i>Menzo Windhouwer and Martin Kersten</i> .....	191
Discovering Association Rules in Temporal Databases / <i>Juan M. Ale and Gustavo H. Rossi</i> .....	195
Document Versioning in Digital Libraries / <i>Mercedes Martínez-González</i> .....	201
E-Government Databases / <i>Catherine Horiuchi</i> .....	206
E-Mail Data Stores / <i>Catherine Horiuchi</i> .....	211
Engineering Information Modeling in Databases / <i>Z.M. Ma</i> .....	216
Ensuring Serializability for Mobile-Client Data Caching / <i>Shin Parker and Zhengxin Chen</i> .....	223
Enterprise Application Integration / <i>Christoph Bussler</i> .....	229
Extended Entity Relationship Modeling / <i>Sikha Bagui</i> .....	233
Extraction-Transformation-Loading Processes / <i>Alkis Simitsis, Panos Vassiliadis, and Timos Sellis</i> .....	240
Free Software and Open Source Databases / <i>Hugo J. Curti</i> .....	246
Fuzzy Database Modeling / <i>Z.M. Ma</i> .....	250
Generic Model Management / <i>Zinovy Diskin and Boris Kadish</i> .....	258

Geometric Quality in Geographic Information / <i>José Francisco Zelasco, Gaspar Porta, and José Luis Fernández Ausinaga</i> .....	266
Hierarchical Architecture of Expert Systems for Database Management / <i>Manjunath R.</i> .....	271
High Quality Conceptual Schemes / <i>Esko Marjomaa</i> .....	276
Information Quality of Databases, The / <i>InduShobha Chengalur-Smith, M. Pamela Neely, and Thomas Tribunella</i> .....	281
Integration of Data Semantics in Heterogeneous Database Federations / <i>H. Balsters</i> .....	286
Integrative Document and Content Management Systems' Architecture / <i>Len Asprey, Rolf Green, and Michael Middleton</i> .....	291
Intension Mining / <i>Héctor Oscar Nigro and Sandra Elizabeth González Císaro</i> .....	298
Kernelized Database Systems Security / <i>Ramzi A. Haraty</i> .....	304
Knowledge Discovery and Geographical Databases / <i>Sami Faiz</i> .....	308
Knowledge Discovery from Databases / <i>Jose Hernandez-Orallo</i> .....	313
Knowledge Management in Tourism / <i>Daniel Xodo and Héctor Oscar Nigro</i> .....	319
Knowledge Mining / <i>Mahesh S. Raisinghani</i> .....	330
Logic Databases and Inconsistency Handling / <i>José A. Alonso-Jiménez, Joaquín Borrego-Díaz, and Antonia M. Chávez-González</i> .....	336
Main Memory Databases / <i>Matthias Meixner</i> .....	341
Managing Inconsistent Databases Using Active Integrity Constraints / <i>Sergio Flesca, Sergio Greco, and Ester Zumpano</i> .....	345
Mathematics of Generic Specifications for Model Management, I / <i>Zinovy Diskin</i> .....	351
Mathematics of Generic Specifications for Model Management, II / <i>Zinovy Diskin</i> .....	359
Metric Databases / <i>Edgar Chávez and Gonzalo Navarro</i> .....	367
Modeling and Querying Temporal Data / <i>Abdullah Uz Tansel</i> .....	373
Moving Objects Databases / <i>M. Andrea Rodríguez-Tastets</i> .....	378
Multilevel Databases / <i>Alban Gabillon</i> .....	383
Multimedia Databases / <i>Mariana Hentea</i> .....	390
Multiparticipant Decision Making and Balanced Scorecard Collaborative / <i>Daniel Xodo</i> .....	395
Natural Language Front-End for a Database / <i>Boris Galitsky</i> .....	403
Normalizing Multimedia Databases / <i>Shi Kuo Chang, Vincenzo Deufemia, and Giuseppe Polese</i> .....	408



Object Modeling of RDBMS Based Applications / <i>Giuseppe Polese, Vincenzo Deufemia, Gennaro Costagliola, and Genny Tortora</i> .....	413
Object-Relational Modeling in the UML / <i>Jaroslav Zendulka</i> .....	421
Online Data Mining / <i>Héctor Oscar Nigro and Sandra Elizabeth González Císaro</i> .....	427
Ontological Assumptions in Information Modeling / <i>John M. Artz</i> .....	433
Ontologies and Their Practical Implementation / <i>Gian Piero Zarri</i> .....	438
Ontology-Based Data Integration / <i>Agustina Buccella, Alejandra Cechich, and Nieves Rodríguez Brisaboa</i> .....	450
Open Source Database Management Systems / <i>Sulayman K. Sowe, Ioannis Samoladas, and Ioannis Stamelos</i> .....	457
Open Source Software and Information Systems on the Web / <i>Antonio Cartelli</i> .....	463
Optimization of Continual Queries / <i>Sharifullah Khan</i> .....	469
Path-Oriented Queries and Tree Inclusion Problems / <i>Yangjun Chen</i> .....	472
Preferred Repairs for Inconsistent Databases / <i>Sergio Greco, Cristina Sirangelo, Irina Trubitsyna, and Ester Zumpano</i> .....	480
Proper Placement of Derived Classes in the Class Hierarchy / <i>Reda Alhajj and Faruk Polat</i> .....	486
Querical Data Networks / <i>Cyrus Shahabi and Farnoush Banaei-Kashani</i> .....	493
Query Operators in Temporal XML Databases / <i>Kjetil Nørnvåg</i> .....	500
Query Processing for RDF Data / <i>Heiner Stuckenschmidt</i> .....	506
Query Processing in Spatial Databases / <i>Antonio Corral and Michael Vassilakopoulos</i> .....	511
Raster Databases / <i>Peter Baumann</i> .....	517
Real-Time Databases / <i>Alejandro Buchmann</i> .....	524
Relational, Object-Oriented and Object-Relational Data Models / <i>Antonio Badia</i> .....	530
Rewriting and Efficient Computation of Bound Disjunctive Datalog Queries / <i>Sergio Greco and Ester Zumpano</i> .....	536
Repairing Inconsistent XML Data with Functional Dependencies / <i>Sergio Flesca, Fillippo Furfaro, Sergio Greco, and Ester Zumpano</i> .....	542
Replication Mechanisms Over a Set of Distributed UDDI Registries / <i>Zakaria Maamar</i> .....	548
Replication Methods and Their Properties / <i>Lars Frank</i> .....	555
Rewriting and Efficient Computation of Bound Disjunctive Datalog Queries / <i>Sergio Greco and Ester Zumpano</i> .....	562
Rhetorical Perspective on Localization and International Outsourcing, A / <i>Kirk St.Amant</i> .....	570
Rough Sets / <i>Zdzislaw Pawlak, Lech Polkowski, and Andrzej Skowron</i> .....	575

Security Controls for Database Technology and Applications / <i>Zoltán Kincses</i> .....	581
Semantic Enrichment of Geographical Databases / <i>Sami Faïz and Khaoula Mahmoudi</i> .....	587
Semantic Information Management / <i>David G. Schwartz and Zvi Schreiber</i> .....	593
Semantically Modeled Enterprise Databases / <i>Cheryl L. Dunn and Severin V. Grabski</i> .....	601
Semistructured Data and its Conceptual Models / <i>Murali Mani and Antonio Badia</i> .....	607
Sensors, Uncertainty Models, and Probabilistic Queries / <i>Reynold Cheng and Sunil Prabhakar</i> .....	613
Service Mechanism Quality for Enhanced Mobile Multimedia Database Query Processing / <i>Yanpu Zhang and Zhengxin Chen</i> .....	619
Set Comparison in Relational Query Languages / <i>Mohammad Dadashzadeh</i> .....	624
Set Valued Attributes / <i>Karthikeyan Ramasamy and Prasad M. Deshpande</i> .....	632
Signature Files and Signature File Construction / <i>Yangjun Chen and Yong Shi</i> .....	638
Similarity Search in Time Series Databases / <i>Maria Kontaki, Apostolos N. Papadopoulos, and Yannis Manolopoulos</i> .....	646
Spatio-Temporal Indexing Techniques / <i>Michael Vassilakopoulos and Antonio Corral</i> .....	652
Storing XML Documents in Databases / <i>Albrecht Schmidt, Stefan Manegold, and Martin Kersten</i> .....	658
Symbolic Objects and Symbolic Data Analysis / <i>Héctor Oscar Nigro and Sandra Elizabeth González Císaro</i> .....	665
Syntactical and Semantical Correctness of Pictorial Queries for GIS / <i>Fernando Ferri and Maurizio Rafanelli</i> .....	671
Temporal Databases / <i>Mahesh S. Raisinghani and Chris Klassen</i> .....	677
Text Categorization / <i>Fabrizio Sebastiani</i> .....	683
Text Databases / <i>Gonzalo Navarro</i> .....	688
Transaction Concurrency Methods / <i>Lars Frank</i> .....	695
Transactional Support for Mobile Databases / <i>Hong Va Leong</i> .....	701
Transformation-Based Database Engineering / <i>Jean-Luc Hainaut</i> .....	707
Ubiquitous Computing and Databases / <i>George Roussos and Michael Zoumboulakis</i> .....	714
Using Semantic Web Tools for Ontologies Construction / <i>Gian Piero Zarri</i> .....	720
Using Views to Query XML Documents / <i>Mario Cannataro, Sophie Cluet, Giuseppe Tradigo, Pierangelo Veltri, and Dan Vodislav</i> .....	729
Vertical Database Design for Scalable Data Mining / <i>William Perrizo, Qiang Ding, Masum Serazi, Taufik Abidin, and Baoying Wang</i> .....	736
XML Multi-Tier Pattern Dissemination System, An / <i>Ashraf Gaffar and Ahmed Seffah</i> .....	740

# Preface

Database research has about 35 years of rich history of productivity which has led to the most relevant and important developments into the software engineering discipline. Naturally, database technologies, architectures and conceptual frameworks have been consolidated in the last decades. Moreover, over the last decade, database management has evolved in such a way that databases have become a key component at the heart of current computing environments and modern information systems. This has provoked a deep impact and significant changes in the way organizations and institutions operate and make their business decisions. It is worthwhile mentioning some facts that have promoted such a growth: the ubiquity of powerful and easy-to-use personal computer database management products, new modeling techniques and tools, most importantly, those based on object oriented thinking, the emergence of client-server processing, the decreasing price of hardware and software, and the imperious need to properly and efficiently manage huge amounts of information.

The history of the database evolution fits into the following paradigm: a problem with data management arises, database technology then copes with the problem in an efficient way and leads users to figure out new problems which in turn feed the evolution of the technology. Perhaps this can be seen as a never-ending cycle of management information needs, new technological advances and successful database software products, which in turn open the gate to new management information needs, and so on. This chain often occurs with any efficient person or program. Their success leads the world to give them more and more tasks!

The design of database applications is a crucial factor in the success of information systems in any organization. Data is one of the most valuable assets to an organization whose validity, consistency and accuracy are vital. A database management systems (DBMS) greatly contributes with these purposes by providing data persistence, efficient access to data, and data integrity guaranteed by integrity constraints. Current database systems offer the SQL query language standard, which has become the query and manipulation language par excellence. By isolating the conceptual schema from the implementation schema, database systems guarantee data independence from storage techniques. Also, by means of the management of users and their privileges, the DBMS can provide secure control access to data. While the control of concurrent access to data is managed through different protocols of transaction scheduling and varied locking techniques, backups and database recovery strategies allow the database recovering after hardware and software failures. All of these areas have opened research fields, exciting challenges, and major technological and conceptual changes in many features through their evolution. This has happened in such a way that database knowledge has become an essential part of the education and research related to computer science.

From another perspective of this fantastic growth of database frameworks, a vast diversity of users, with their areas of interest, particular application requirements and own technological needs, have become interested in databases. In recent years, these facts have promoted a general tendency towards new and original database research areas and practice fields, framed into new approaches. Other new and powerful issues of database research are programming languages, active databases, temporal databases, spatial databases, multimedia databases, and databases and the Web.

The growth of database effectiveness was accompanied by a huge increase of users and user profiles. The need of users for information sources has naturally and extensively grown. We believe this encyclopedia will help to close the breach between both aspects.

The *Encyclopedia of Database Technologies and Applications* is a collaborative effort that addresses the evolution of database management, technologies and applications along with the progress and endeavors of new research areas.

Leading specialists in each area, researchers with profuse publications on the topics covered by this volume, practitioners with a vast experience in the development of database systems in organizational environments, and teachers with accumulated experience teaching graduate and undergraduate courses have contributed articles on their field of expertise to this encyclopedia.

The potential audience of this encyclopedia is widely diverse. This encyclopedia is intended for computing students who have some knowledge on databases, for teachers giving not only introductory but advanced courses on databases, for researchers with concerns about specific concepts in their area of interest, and for practitioners facing database implementation choices. Inexperienced readers and students will also find in this encyclopedia an invaluable basis to increase their knowledge about databases, in addition to the main issues and applications related to databases. Experienced teachers will discover a comprehensive compendium of teaching resources. On the other hand, this database encyclopedia is also a valuable reference book for professionals, designers, application programmers, and database practitioners. This important new publication brings together a discussion of a wide spectrum of database issues, giving researchers, scholars, students, and professionals the access to the latest knowledge related to database science, technology and the trends in this promising and continuously evolving field. We can say that the main endeavor of this encyclopedia has been to grant access to essential core material for education, research, and practice on database systems.

This encyclopedia is a compilation of entries with a standardized structure and style sorted by title. It is broader than most database books since it provides a comprehensive and balanced treatment of the most important issues, concepts, languages, and definitions of the core terms in each subject. It recommends concrete references to current technology and includes profuse links to relevant literature on each topic. It offers a sound grounding in the foundations of database technology and the state of the art and also provides solid ideas about current trends and evidence on how this field is likely to develop in the near future. Many articles offer an analytical approach, so that the concepts presented can be applied not only to the wide variety of actual databases but also can be used as a fundamental stone to build future systems.

The areas covered in this encyclopedia are under an exceptionally fast development and spreading. For this reason, each article of this volume includes a list with the Key Terms and concepts of each topic along with their definitions. The choice of such terms has been a great success of our contributors.

Topics covered by this encyclopedia include the analysis of information requirements, the development of conceptual data models, the improvement of relational and object oriented schemas, SQL and XML concerns, database integrity, database performance and optimization, transactions, backup and recovery strategies, file structures and indexing, and the development of challenging database applications. Many articles have numerous examples on how to apply the material.

## ORGANIZATION OF THIS ENCYCLOPEDIA

This encyclopedia has been organized alphabetically by title, in order the reader to find quickly the article of his interest.

This preface introduces the main research areas exposed. It is not intended to detail all the issues and results, but to give an initial orientation of the work introduced in the research articles. A short description of each area covered follows:

*Knowledge management* is generally seen as a problem of treating the capture, organization, and retrieval of information. Related to this area, the design of knowledge management systems including in that design the human and social factors, constitute a novel and promising research discipline that has been covered in few articles. The reader will find in them particular applications of this approach.

*Data integration* is the subject of various articles. As the storing and retrieving of data are not enough in newest organizational environments, new approaches to data integration have been emerged. One of them is the *data warehousing* approach. Conventional and current database systems are (and typically have been) designed and optimized for On-Line Transaction Processing (OLTP). In this sense, the databases can manage huge quantities of transactions executing concurrently, modifying or reading generally small and changeable amounts of data. However, recently applications in the context of the management of a complex enterprise require a far-reaching, overall view of all aspects of its activities. The core characteristic of the *data warehousing* approach is that the data of interest come from heterogeneous sources. Since those data are often of a different format and complexity; for instance they can come from different databases, installed from different platforms, or simply from online multimedia data stores, Web pages, spreadsheets, or even conventional files. This poses not only difficulties inherent to its access, but new challenges on the way to provide a uniform, integrated and transparent access to such heterogeneous and distributed information sources. *Federated database systems* are one of the most recent advances that allow distributed databases and other data sources to be virtually integrated.

Related to this new trend, special attention has been paid to the analysis of operational data in order to discover tendencies, singularities, patterns of behavior and perhaps some anomalies. Different analytical tools, called *decision support systems* (DSS) and *data mining*, enable the analysts to make better decisions. All of these related issues have promoted numerous research efforts whose main results can be read in several articles of this encyclopedia.

Several entries have been devoted to *object-oriented database management system* (OODBMS) issues. These systems have generated developments and challenges in the database community, mainly promoted by the genuine requirement of support to the management of data, which is not very well carried out by the relational technology. The advances accomplished through this approach are well-known and are giving rise to the so called “advanced application areas”. Although the market for OODBMSs is still small, this technology has rapidly gained maturity and has notoriously influenced other paradigms. This scenario has given birth to a variety of applications, which are in constant growth and have foreseen the convenience of using this technology.

The World Wide Web can be seen as a database, although it is inherent chaos, without enforcing any condition to the sites members of this net. For this reason, the challenge is to create a structure inside the chaos by means of special queries. Many articles provide an excellent snapshot of current research, development activities, and trends in the area of *Web or Internet databases*. The Web as a huge database, and its implications regarding the progressive adaptation of core database techniques to Web usage, are quickly evolving and have created a massive demand for Web-based information systems, posing interesting challenges for researchers and practitioners.

A commencement of organization has been accomplished through the consolidation of the eXtensible Markup Language (XML) as a new standard adopted by the World Wide Web Consortium (W3C). It has facilitated the management of electronic data, mainly due to emerging database system’s current application needs interacting with each other over the Web. Nowadays the researchers talk about the Second-Generation Web. The conjunction of XML and related technologies have resulted in the so-called *semantic Web*, in such a way that the computers will be able to deal with the meaning (“semantics”) of Web data. Several articles are devoted to these topics, covering the state-of-the-art in databases for the Internet era, in addition to the basic material of the main issues related to databases accessed through and interacting on the World Wide Web. These articles conform a strong starting point for practitioners interested in the development of database applications on the Web.

Nowadays, the profound knowledge of the relational model, the evolution to the object-relational model, object-oriented, and deductive models are a core need to effectively face the development of most industrial applications. In this regard, the articles which focus on the related areas of *modeling* and *CASE tools* reflect a wide range of current research and practice areas, given that the quality of the results obtained in the early stages of design are a main factor of the success of an information system. The discussion of design techniques covers topics, which range from the introduction of the concepts of classic models to a novel view on modern metadata management problems. Research on conceptual modeling, its state of the art, modern database paradigms, requirements engineering, CASE tools implementation, and comparison are extensively explored.

Data models’ evolution over time, due to the changes of the user requirements, is a current concern and the subject of articles mainly dedicated to the case of databases. Versioning is one of the useful approaches to manage the evolution of conventional and non-conventional databases. This encyclopedia offers articles providing insight on *database evolution* and *versioning* areas.

*Query languages* constitute a key piece of data models. Some topics covered extensively are the use of database front-ends to support a useful way to translate queries from natural language to SQL, the use of specific algorithms to provide efficient *query processing* in the case of static time series of data, the description and overcoming of the weaknesses of *relational query languages* in expressing set comparison queries, *query optimization* as an active area of research recently renewed due to the appearance of data warehousing, and the analysis of different approaches for the efficient *evaluation of queries*.

*Database integrity* is another main subject addressed in this encyclopedia. Articles provide not only a background on basic knowledge about this concern but innovative perspectives and implementations.

Several articles cover *Active database systems*, concepts on active rules and the way these rules are supported in representative relational systems, discussions about how active rules can be generated for integrity maintenance and tested for termination, as well as current applications and trends related to this challenging paradigm.

*Temporal issues* are also addressed in this volume. Temporal data are data, which represent some aspect of time, incorporating this concept to conventional databases. Some implementation issues and approaches to implement a temporal DBMS as a layer on top of an existing non-temporal DBMS, aspects such as the adaptation of the framework for concurrent processing of operations ensuring the ACID properties of transactions, and the extension of the SQL language to incorporate temporal issues are some of the topics explored in contributed articles.

Multimedia technologies are gaining popularity day after day and recently they constitute an important component of the study of computer graphics, image processing, real-time systems, information retrieval, computer vision, and many other related fields. Modern database systems have enhanced the capabilities of traditional database systems to cope with any kind of data, including text, image, audio, and video, and to provide access to these data via SQL-like query languages. To make such immense amount of data available, a vast research effort is being developed focusing on design models of *multimedia databases* and novel methodologies to provide efficient image indexing and retrieval. This encyclopedia includes articles focusing this research field.

*Digital libraries* are systems that contain organized collections of objects, in such a way that they resemble traditional libraries containing paper documents. Generally, most of the information contained in a digital library consists in documents. Throughout the last decades the production, storing, retrieving, and management of digital documents has increased explosively, provoking an increased need for solutions for those activities.

From a different perspective, the huge amount of text data available in current applications poses the challenge of effectively searching them for queries of interest. A text database is a system that maintains a text collection and provides fast and accurate access to it. As traditional database technologies are not well suited to handle *text databases*, this novel approach faces the challenge of solving the need of specific technology to manage such kind of data. Digital libraries and text databases are the main subject of several articles.

Fundamental issues of database technology such as transactions, concurrency control, reliability, and recovery are explored into a set of entries, with emphasis on *distribution*, *replication* and *parallelism*. The purpose of these writings is to offer the reader not only the state of the art subjects but to disseminate the knowledge and challenges associated with the scale-up of distributed systems.

Another core concept in the database area is *security*. Conventional approaches address discretionary security by controlling access privileges and modes of data users. On the other hand, mandatory security provides a higher level of assurance by restricting the access to data items to cleared database users. This last approach is adequate for such commercial and military applications requiring a multilevel secure database management system. Relevant authors explore these topics in a set of articles.

In previous years, *Free Software* and *Open Source databases* (OSDBMS) have evolved in a remarkable way, becoming a valuable alternative to the commercial databases. The articles devoted to this topic not only focus on the tendencies outlined by OSDBMS, but also on the most pertinent licenses and in the projects of the database and applications developed by the Free/Open Source Software Development community and on the central role played by this kind of software.

*GIS (Geographic Information Systems)* allows users to model, store, and analyze information describing physical properties of the real world. Spatial data coming from maps, digital images, routes, field elevations, and data coming from a diversity of censuses, economic studies, market indicators, etc. are the kind of material the systems deal with. Many articles address issues such as data quality, query languages, and integrity within this framework.

The articles which focus on other advanced database applications deal with information retrieval by means of the creation and updating of *database indexes* and their application to newest kind of non-conventional data. These articles also focus on specialized databases such as *biometric databases*, which research basic problems related with pattern recognition, biometric authentication and intelligent surveillance, *bioinformatics* as a recent discipline into the information science, which concerns the creation and maintenance of databases of biological information, to store, extract, analyze, and the use of biological information.

The emergence of powerful portable computers and the development of fast reliable networks have promoted the birth and growing of a new dimension in data processing and communication: *mobile computing*. Mobility and portability have created a new repertory of applications and have produced a deep impact in the data management field. In this sense, *mobile database systems* have emerged as a paradigm bringing support to transactions provided by mobile clients and creating a broad and promising research field in the past decade. Entries exploring mobile databases focus on data dissemination over limited bandwidth channels, location-dependent querying of data, and concurrency control and recovery mechanisms.

In summary, this encyclopedia offers a unique and exciting opportunity to find out, not only about the state of the art of fundamental database concepts, but about current trends, novel technology, and challenging applications to meet the needs of current data management systems.

# Acknowledgments

We would like to thank Mehdi Khosrow-Pour for this opportunity. It has been a challenge and a great learning experience. We would also like to acknowledge the Idea Group staff for their support and advice, mainly to Jan Travers, Sara Reed and Renée Davies, for their guidance, professional support, and continuous stimulus.

We also want to thank all of the people who assisted us in the reviewing process. First, thanks to the encyclopedia authors, which have served as referees, doing an invaluable job refereeing other articles by providing constructive and comprehensive reviews.

Thanks also to the external reviewers who had provided many significant and constructive comments:

From Universidad Nacional del Centro, Argentina

Dr. Eduardo Caselli

MSc. Mariana del Fresno

D.E.A. Liliana Favre

MSc. Carmen Leonardi

MSc. Virginia Mauco

From LIFIA, Universidad Nacional de La Plata, BA, Argentina

Dr. Silvia Gordillo

Dr. Claudia Pons

From Universidad Nacional de La Matanza, Argentina.

Professor Alejandro Oliveros

From the Agricultural University of Athens, Greece

Dr. Nikos Lorentzos.

From the Department of Computer Sciences, University of Tampere, Finland.

Professor Pirkko Nykänen

From University of Rhode Island, USA.

Dr. Jounghae Bang

From the Universidad Nacional de San Luis, Argentina;

MSc. Nora Reyes

From the Universidad Argentina de la Empresa, Argentina;

Dr. Ricardo Orozco

From University of North Carolina, USA.

Dr. Shannon Schelin



From University of Jyväskylä, Finland.  
Dr. Virpi Lyytikäinen

From Universidad Autónoma de Madrid, Spain.  
Dr. Roberto Moriyón

From Kettering University, USA.  
Dr. Andrew S. Borchers

From Universidad Nacional de Belgrano, Argentina  
Professor Graciela Hadad

From Università of Salerno, Italy.  
Dr. Monica Sebillio

From IT Transfer Office (ITO) at Darmstadt University of Technology, Germany.  
Dr. Jan Steffan

From Universidad de Castilla-La Mancha, Spain.  
Dr. Eduardo Fernández-Medina Patón  
Dr. Mario Piattini

From University of Sharjah, UAE  
Dr. Zaher Al Aghbari

From Kyushu University, Japan  
Dr. Akifumi Makinouchi

From Jawaharlal Nehru Technological University College of Engineering, Kukatpally, Hyderabad  
Dr Chandra Sekharaiah

From Integral Science Data Centre, Switzerland.  
Dr. Mohamed Tahar Meharga

We are grateful to the authors for their insights and excellent contributions to this encyclopedia.

The editors would like to acknowledge the help of all people involved in the collation and review process of this encyclopedia, without whose support the project could not have been satisfactorily completed.

In closing, we would like to express our deep gratitude to the many colleagues and friends who have encouraged us to face this project and who have animated us in those long work days.

*Laura Celia Rivero*  
*Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina*

*Jorge Horacio Doorn*  
*Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina*

*Viviana Elizabeth Ferraggine*  
*Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina*

## About the Editors

**Laura C. Rivero** received her BS degree in systems analysis from the Universidad Nacional del Centro (UNCPBA) (Argentina), in 1979. She is a professor in the Department of Computer Science and Systems of the UNCPBA and a doctoral student in computer science at the Universidad Nacional de La Plata. Her research interests include data structures, extended relational database systems and database design, integrity and reengineering. She has published several articles in refereed journals and international conferences in the above areas.

**Jorge H. Doorn** is full professor in the Computer Science Department at the Universidad Nacional del Centro (UNCPBA) (Argentina), since 1989. He has wide experience in actual industrial applications. He has been project leader in several projects and currently he is the leader of the database research team in the Computer Science and Systems Department. His research interests include requirements engineering, database systems and signal processing. He has edited three compiled books.

**Viviana E. Ferraggine** received her BS degree in systems engineering from the Universidad Nacional del Centro (UNCPBA) (Argentina), in 1997. She is currently an auxiliary assistant with the Department of Computer Science and Systems and a master student in computer science at the Universidad Nacional de La Plata. Her research interests include database systems, conceptual modeling and reengineering and data structures. Currently she is coordinating and teaching in courses on database systems.



# Active Database Management Systems

Mariano A. Cilia

Technische Universität Darmstadt, Germany and UNICEN, Argentina

## INTRODUCTION

As it is well known, business requirements are changing faster than applications can be created and/or modified. Most of these requirements are in the form of or are related to business rules. Business rules are precise statements that describe, constrain and control the structure, operations and strategy of a business. They may be thought of as small pieces of knowledge about a business domain. They offer a way of encapsulating business semantics and making them explicit in the same way that databases enable the separation of data from application programs.

Traditionally, business rules have been scattered, hard-coded and replicated by different applications. The lack of a formal approach to the management of business rules and a standard business rule language has made it virtually impossible to create, modify and manage business rules in a flexible way. As a result, it has been difficult to adapt applications to new requirements quickly.

Isolating the business rules from the application code enables developers to easily find and modify the pertinent rule(s) when a policy change is needed or when application requirements change. This makes it possible to quickly change rules without modifying the rest of the application code, thereby enhancing maintainability.

In the last decade, one of the trends in database technology has focused on extending conventional database systems (DBMSs) to enhance their functionality and to accommodate more advanced applications. One of these enhancements was extending database systems with powerful rule-processing capabilities. These capabilities can be divided into two classes: *deductive*, in which logic-programming-style rules are used to provide a more powerful user interface than that provided by most database query languages (Ceri, Gottlob, & Tanca, 1990), and *active*, where production-style rules are used to provide automatic execution of predefined operations in response to the occurrence of certain events (Act-Net Consortium, 1996; Dayal, Buchmann & McCarthy, 1988). The latter is particularly appropriate for enforcing business rules as it has been demonstrated in Ceri and Widom (1996) and Paton (1999). Database systems enhanced with active capabilities are known as *active databases systems*, or aDBMSs for short.

By means of active database systems, general integrity constraints encoded in applications have been moved

into the database system in the form of rules. These rules go beyond key or referential integrity constraints. Active databases support the specification and monitoring of general constraints (rules), they provide flexibility with respect to the time of constraint checking, and they provide execution of compensating actions to rectify a constraint violation without rolling back the involved transaction. Additionally, support for external events and actions were introduced mostly to satisfy the requirements of monitoring applications.

As a consequence, applications sharing the same database system and data model can also share business rules. In this way, the business knowledge that was dispersed in many applications in the form of programming code is now represented in the form of rules and managed in a centralized way. Consequently, when business rules change, only those affected rules must be modified in the aDBMS.

## BACKGROUND

Historically, production rules were used first to provide automatic reaction functionality. Production rules are Condition-Action rules that do not break out the triggering event explicitly. Instead, they implement a polling-style evaluation of all conditions. In contrast, Event-Condition-Action (ECA) rules explicitly define triggering events, and conditions are evaluated only if the triggering event was signaled. Active databases adopted this event-driven approach to avoid unnecessary and resource-intensive polling in monitoring database changes and applications.

In active relational databases, events were modeled as changes of the state of the database, i.e., *insert*, *delete* and *update* operations on tables (Hanson, 1989; Stonebraker, Jhingran, Goh, & Potamianos, 1990). This basic functionality is common fare in commercial DBMSs today. In object-oriented systems, more general primitive events were defined: temporal events, both absolute and relative; method invocation events; and user-defined events (Dayal & Blaustein, 1988; Gatzju & Dittrich, 1993; Zimmermann & Buchmann, 1999).

In addition to these primitive events, more complex situations that correlate, aggregate or combine events can be defined. This is done by using an event algebra

(Zimmer & Unland, 1999) that allows the definition of so-called *composite* or *complex events*.

By means of a rule definition language (RDL), ECA rules are specified. This language provides constructors for the definition of rules, events, conditions and actions. Once rules are specified with an RDL, the rule base should be checked according to the following properties: termination, confluence and observably deterministic behavior (Aiken, Widom, & Hellerstein, 1992). For instance, termination analysis ensures that the current rule base is safe and correct with respect to the intended reactive behavior. In general, these properties cannot be proven automatically, but an analyzer might assist in detecting inconsistencies. After this verification step, rules are ready to be processed.

## ECA RULE PROCESSING

Rule execution semantics prescribe how an active system behaves once a set of rules has been defined. Rule execution behavior can be quite complex, but we restrict ourselves to describing only essential aspects here. For a more detailed description, see Act-Net Consortium (1996) and Widom and Ceri (1996).

All begins with event instances signaled by event sources that feed the complex event detector, which selects and consumes these events. *Consumption modes* (Chakravarthy, Krishnaprasad, Anwar, & Kim, 1994) determine which of these event instances are considered for firing rules. The two most common modes are *recent* and *chronicle*. In the former, the most recent event occurrences are used, while in the latter, the oldest event occurrences are consumed. Notice that in both cases a temporal order of event occurrences is required. Different consumption modes may be required by different application classes.

Usually there are specific points in time at which rules may be processed during the execution of an active system. The *rule processing granularity* specifies how often these points occur. For example, the finest granularity is “always,” which means that rules are processed as soon as any rule’s triggering event occurs. If we consider the database context, rules may be processed after the occurrence of database operations (small), data manipulation statements (medium), or transactions (coarse).

At granularity cycles and only if rules were triggered, the *rule processing algorithm* is invoked. If more than one rule was triggered, it may be necessary to select one after the other from this set. This process of *rule selection* is known as *conflict resolution*, where basically three strategies can be applied: (a) one rule is selected from the fireable pool, and after rule execution, the set is deter-

mined again; (b) sequential execution of all rules in an evaluation cycle; and (c) parallel execution of all rules in an evaluation cycle.

After rules are selected, their corresponding conditions are evaluated. Notice that conditions can be expressed as predicates on database states using a query language, and also external method invocations can be used. The specification of conditions can involve variables that will be bound at runtime with the content of triggering events. If a condition evaluates to true, then the action associated with this rule must be executed. Actions can be any sequence of operations on or outside of a database. These operations can include attributes of the triggering event.

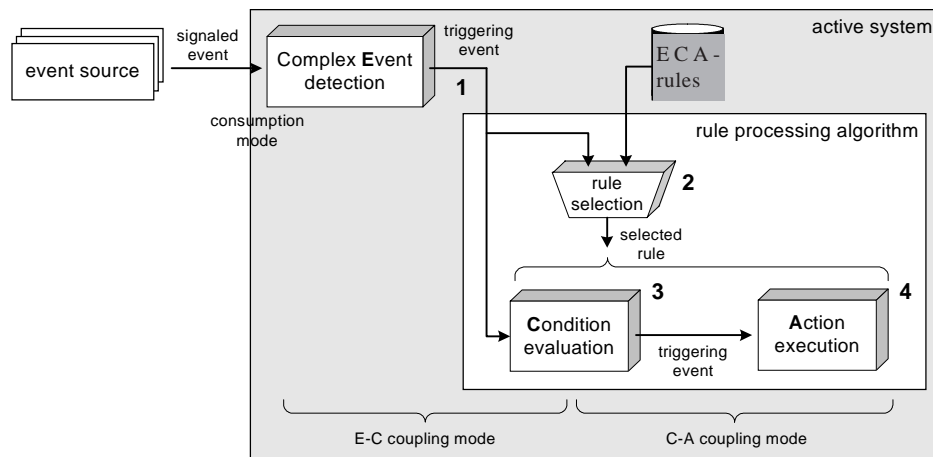
For some applications it may be useful to delay the evaluation of a triggered rule’s condition or the execution of its action until the end of the transaction, or it may be useful to evaluate a triggered rule’s condition or execute its action in a separate transaction. These possibilities lead to the notion of *coupling modes*. One coupling mode can specify the transactional relationship between a rule’s triggering event and the evaluation of its condition while another coupling mode can specify the transactional relationship between a rule’s condition evaluation and the execution of its action. The originally proposed coupling modes are immediate, deferred and decoupled (Dayal & Blaustein, 1988).

To sum up, an ECA-rule-processing mechanism can be formulated as a sequence of four steps (as illustrated in Figure 1):

1. **Complex event detection:** It selects instances of signaled events in order to detect specified situations of interest where various event occurrences may be involved. As a result, these picked instances are bound together and signaled as a (single) complex event.
2. **Rule selection:** According to the triggering events, fireable rules are selected. If the selection is multiple, a conflict resolution policy must be applied.
3. **Condition evaluation:** Selected rules receive the triggering event as a parameter, thus allowing the condition evaluation code to access event content. Transaction dependencies between event detection and the evaluation of a rule’s condition are specified using Event-Condition coupling modes.
4. **Action execution:** If the rule’s condition evaluates to true, the corresponding action is executed taking the triggering event as a parameter. Condition-action dependencies are specified similarly to Step 3.

It should be noticed that Step 1 (complex event detection) can be skipped if rules involve primitive events only.

Figure 1. Schematic view of the ECA-rule-processing mechanism



## TOOLS

Several tools are required in order to adequately support the active functionality paradigm (Act-Net Consortium, 1996). For instance, a *rule browser* makes possible the inspection of the rule base; a *rule design assistance* supports the process of creation of new rules and is complemented with a *rule analyzer* that checks for a consistent rule base; and a *rule debugger* monitors the execution of rules.

## FUTURE TRENDS

Modern large-scale applications, such as e-commerce, enterprise application integration (EAI), Internet or intranet applications, and pervasive and ubiquitous computing, can benefit from this technology, but they impose new requirements. In these applications, integration of different subsystems, collaboration with partners' applications or interaction with sensor devices is of particular interest.

It must be noticed that in this context, events and data are coming from diverse sources, and the execution of actions and evaluation of conditions may be performed on different systems. Furthermore, events, conditions and actions may not be necessarily directly related to database operations. This leads to the question of why a full-fledged database system is required when only active functionality and some services of a DBMS are needed.

The current trend in the application space is moving away from tightly coupled systems and towards systems of loosely coupled, dynamically bound components. In such a context, it seems reasonable to move required active functionality out of the active database system by offering a flexible service that runs decoupled from the database. It

can be combined in many different ways and used in a variety of environments. For this, a component-based architecture seems to be appropriate (Collet, Vargas-Solar, & Graziotin-Ribeiro, 1998; Gatzju, Koschel, von Buetzingsloewen, & Fritschi, 1998;), in which an *active functionality service* can be seen as a combination of other components, like complex event detection, condition evaluation and action execution. Thus, components can be combined and configured according to the required functionality, as proposed by the unbundling approach in the context of aDBMSs (Gatzju et al.) or by using a service-oriented architecture (SOA) as described in Cilia, Bornhövd, and Buchmann (2001).

## CONCLUSION

Active databases support the specification of business rules in the form of ECA rules. The business knowledge that was dispersed in many applications in the form of programming code is now represented in the form of rules and managed in a separated way. This facilitates the adaptation to new requirements and improves the maintainability of systems.

As mentioned above, the current trend of active functionality technology shows a shift towards loosely coupled and distributed systems and the availability of such functionality as a service.

## REFERENCES

Act-Net Consortium. (1996). The active database management system manifesto: A rulebase of ADBMS features. *ACM SIGMOD Record*, 25(3), 40-49.

- Aiken, A., Widom, J., & Hellerstein, J. (1992). Behavior of database production rules: Termination, confluence, and observable determinism. *Proceedings of ACM International Conference on Management of Data (SIGMOD)*, 59-68.
- Ceri, S., Gottlob, G., & Tanca, L. (1990). *Logic programming and databases*. Springer.
- Ceri, S., & Widom, J. (1996). Applications of active databases. In J. Widom & S. Ceri (Eds.), *Active database systems* (pp. 259-291). Springer.
- Chakravarthy, S., Krishnaprasad, V., Anwar, E., & Kim, S. (1994). Composite events for active databases: Semantics, contexts and detection. *Proceedings of the International Conference on Very Large Databases (VLDB)* (pp. 606-617).
- Cilia, M., Bornhövd, C., & Buchmann, A. (2001). Moving active functionality from centralized to open distributed heterogeneous environments. In *Lecture notes in computer science: Vol. 2172. Proceedings of the International Conference on Cooperative Information Systems, CoopIS* (pp. 195-210).
- Collet, C., Vargas-Solar, C., & Grazziotin-Ribeiro, H. (1998). Towards a semantic event service for distributed active database applications. In *Lecture notes in computer science: Vol. 1460. Proceedings of the International Conference on Databases and Expert Systems Applications, DEXA* (pp. 16-27).
- Dayal, U., Blaustein, B., Buchmann, A., Chakravarthy, S., Hsu, M., Ledin, R., et al. (1988). The HiPAC project: Combining active databases and timing constraints. *ACM SIGMOD Record*, 17(1), 51-70.
- Dayal, U., Buchmann, A., & McCarthy, D. (1988). Rules are objects too. In *Lecture notes in computer science: Vol. 334. Proceedings of the 2nd International Workshop on Object-Oriented Database Systems* (pp. 129-143).
- Gatziau, S., Koschel, A., von Buetzingsloewen, G., & Fritschi, H. (1998). Unbundling active functionality. *ACM SIGMOD Record*, 27(1), 35-40.
- Gatziau, S., & Dittrich, K. R. (1993). Events in an active object-oriented database system. *Proceedings of the 1st International Workshop on Rules in Database Systems (RIDS'93)* (pp. 23-29).
- Hanson, E. (1989). An initial report on the design of Ariel: A DBMS with an integrated production rule system. *ACM SIGMOD Record*, 18(3), 12-19.
- Paton, N. (Ed.). (1999). *Active rules in database systems*. Springer.
- Stonebraker, M., Jhingran A., Goh, J., & Potamianos, S. (1990). On rules, procedures, caching and views in database systems. *Proceedings of the 1990 ACM International Conference on Management of Data (SIGMOD)* (pp. 281-290).
- Widom, J., & Ceri, S. (Eds.). (1996). *Active database systems: Triggers and rules for advanced database processing*. Morgan Kaufmann.
- Zimmer, D., & Unland, R. (1999). On the semantics of complex events in active database management systems. *Proceedings of the 15th International Conference on Data Engineering (ICDE'99)* (pp. 392-399).
- Zimmermann, J., & Buchmann, A. (1999). REACH. In N. Paton (Ed.), *Active rules in database systems* (pp. 263-277). Springer.

## KEY TERMS

**Business Rules:** They are precise statements that describe, constrain and control the structure, operations and strategy of a business. They may be thought of as small pieces of knowledge about a business domain.

**Consumption Mode:** It determines which of these event instances are considered for firing rules. The two most common modes are recent and chronicle.

**Coupling Mode:** It specifies the transactional relationship between a rule's triggering event, the evaluation of its condition and the execution of its action.

**ECA Rule:** It is a (business) rule expressed by means of an event, a condition and an action.

**Event:** It is an occurrence of a happening of interest (also known as primitive event). If the event involves correlation or aggregation of happenings then it is called a complex or composite event.

**Event Algebra:** Composite events are expressed using an event algebra. Such algebras require an order function between events to apply event operators (e.g., sequence) or to consume events.

**Rule Base:** It is a set of ECA rules. Once this set is defined, the aDBMS monitors for relevant events. The rule base can be modified (new rules can be added, or existent rules can be modified or deleted) over time.

**Rule Definition Language (RDL):** Set of constructs for the definition of rules, events, conditions and actions.



# Active Federated Database Systems

Genoveva Vargas-Solar

National Centre for Scientific Research (CNRS), France

A

## INTRODUCTION

Database management systems (DBMS) are becoming part of environments composed of large-scale distributed heterogeneous and networks of autonomous, loosely coupled components. In particular, federated database management systems (FDBMS) can be seen as networks that integrate a number of pre-existing autonomous DBMS which can be homogeneous or heterogeneous. They can use different underlying data models, data definition and manipulation facilities, transaction management, and concurrency control mechanisms. DBMS in the federation can be integrated by a mediator providing a unified view of data: a global schema, a global query language, a global catalogue, and a global transaction manager. The underlying transaction model considers, in general, a set of transactions synchronized by a global transaction. Synchronization is achieved using protocols such as the Two-Phase Commit protocol. FDBMS applications are built upon this global environment, and they interact with the mediator to execute global database operations (i.e., operations that can concern various DBMS in the federation).

In order to illustrate intuitively the use of FDBMS, consider a financial context where a shareholder has two bank accounts in Mexico and in France managed by database applications. This person needs to execute banking operations either accessing the accounts independently or making applications cooperate to have a global view of his/her financial situation. Clearly, an FDBMS application would fulfill these requirements, letting to execute transparently banking global (e.g., consult of the global credit) and local operations (e.g., withdraw 1000 pesos from the account in Mexico). Assume now that shareholders need specific operations to be executed timely when certain conditions come up:

```

WHEN the dollar price changes in France,
F      my account in France has more than
      100000 euros
THEN  send an e-mail to advise me to buy
      dollars.

WHEN  money is withdrawn from my bank
F      accounts,
      my global credit is less than 1000
      pesos
THEN  abort the operation.

```

Even if some DBMS provide some active capabilities (triggers), federated database systems are often limited when considering the encoding of the behavior of data and the reaction of the system(s) to data changes. Operations are executed after explicit request submitted by a user or an application. These systems may be referred as passive FDBMS, in opposition to active FDBMS that are able to execute automatically predefined actions in reaction to specific events when some conditions are satisfied.

As in active DBMS (Paton, 1998), the major idea in active FDBMS is to add a reactive mechanism as ECA rules. Rules are processed in a specific environment defined as an active FDBMS application including global transactions. In such environments, possible uses of active rules are numerous, for instance, are view maintenance, global schema updates, verification and validation of global integrity constraints, notification, application, component integration and cooperation, and so forth.

The use of rules in FDBMS applications implies at least three challenges. First, the inherent heterogeneity of the system imposes the need of a flexible rule execution model adaptable to the characteristics of the participating DBMS. Second, the active mechanism must deal with the autonomy of both the global system itself and the participating DBMS. In an FDBMS, DBMS can keep their communication and execution autonomy. Thus, they may share or not control information, and they may continue to control their execution at any moment, independently of the federation. They can commit or abort local transactions at any time, and this can affect the execution of global operations. Third is events management stemming from different contexts. Communication protocols are needed to observe events from their sources (DBMS) and signal them to consumers (rules). Events are messages containing information about the federation and its components. Therefore, they should be processed respecting information consistency, legacy, and performance needs.

For providing active capabilities within FDBMS, it is necessary to go beyond what has been proposed and developed in the context of the active databases domain (Chakravarthy, Le & Dasari, 1998). This article proposes an event service and a rule service that cooperate to execute active rules within an FDBMS.

## THE SERVICES APPROACH

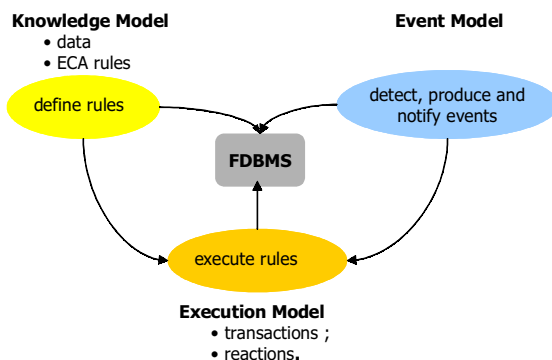
Active FDBMS require a federation-wide mechanism for event handling and reaction execution. In such a context, it must be possible to detect events to make them visible to other components of the federation. Events can be either observed or raised by each participating DBMS. It must also be possible to couple the execution of actions with the execution of FDBMS applications.

From the wide variety of proposals, distribution has not been widely introduced in active database environments. Furthermore, the experience of research works shows that active capabilities are potentially useful and pertinent to answer to non-database applications. However, the architecture of these systems does not allow providing active functionalities able to fulfill the requirements of every application. Therefore, we provide active capabilities as services instead of a monolithic system. Services are independent components, and they are the core of an extensible mechanism. The problem is how to unbundle the mechanism so that a full-functioning and lean active mechanism remains. Figure 1 shows processes that have to cooperate for getting an active FDBMS, each of them based on a model.

The knowledge model describes how to define, represent, and administrate data and rules. The FDBMS data model defines the data structure and the operations that can be executed over data. The rule data model defines the Event, Condition, and Action parts of the rule. Conditions and actions are specified considering the FDBMS data model. Hence, they may implicitly concern several DBMS in the federation.

Event definition is supported by an event model. It specifies the set of event types representing the significant situations that have to be observed across and within the FDBMS (event type model) and the way such events have to be detected, composed, and notified (event management model).

Figure 1. Active FDBMS



An FDBMS application is executed according to the execution model of the system (e.g., global transactions). Similarly, rules are executed according to an execution model that characterizes the coupling of event consumption, condition evaluation, and action execution within and across FDBMS or global transactions.

Implementing the above models must not demand the complete re-definition of the federation and applications. Therefore, we first isolated the active mechanism from the FDBMS control. Then, we isolated the event mechanism from the rule execution one to specify two independent services.

The event service detects complex situations within clients (DBMS, applications) and notifies them automatically to other clients. The rule service executes asynchronous reactions to events. The technical challenge of this approach is the amalgamation of communication, database, and active aspects (Bützingsloewen et al., 1998).

The event service administrates event managers that detect, produce, and notify events. The FDBMS and application execution are monitored to detect and produce (i.e., recognize, order, and compose) events described by rules. Then, events are delivered to the rule service under different modes and communication protocols depending on rule execution models. The rule service is responsible for rule execution that consists of two phases:

- The planning phase that builds an execution plan according to different strategies (Coupaye & Collet, 1998).
- The execution of one rule: the evaluation of the Condition part and the execution of the Action, if the Condition is satisfied.

A parametric interface to specify interaction among the services of the FDBMS has been proposed in Collet, Vargas-Solar, and Graziotin-Ribeiro (2000) and Graziotin-Ribeiro (1999).

## EVENT SERVICE

The event service defines a framework for specifying and generating (implementing) event managers. It supports two meta-models for specifying event managers (i.e., defining their event type system and their event management model).

### Event Type Description

The Event Type Meta-Model (Collet et al., 1998; Vargas-Solar, 2000; Vargas-Solar & Collet, 2002) provides con-

cepts for describing primitive event types (names and contexts) that concern specific application domains, as well as semantics of composition operators. An event type is an expression describing operations executed by a particular source, produced over a validity time interval. The validity time interval defines the interval during which events of the actual type can be recognized. Thus, events are instances of an event type that can be significant for consumers (e.g., the rule service).

An expression can describe primitive or composite events. In FDBMS, primitive event types represent database operations, temporal- and user-defined events. Composite event types characterize events produced by applying operators such as sequence, conjunction, disjunction, negation, parallel, and so forth to other events.

Event types are instances of the meta-model and are characterized by an instant of detection, a name, an algebraic expression, a validity time interval, a context, and a mask. Types are defined in a parametric way. In our example, assume that a service delivers financial information such as the variation of the dollar price. The event type the dollar price changes  $E_1$  can be defined as follows:

```
<instant> (UpdateDollar, [sv,ev]) [with delta] [where mask].
```

The type represents all the possible events representing the fact that the dollar price changes; this is why some of the attributes of the type are not instantiated. The type just specifies that all events will have an instant of production, an eventually associated information about their context [with delta] and a filter [where mask].

When a rule adopts an event type, it specifies the instant of detection (before, after, or duration based) and it instantiates the validity time interval, delta, and mask. Valid values are those specified in the domains of the corresponding event management model.

## Event Management

The Event Management Meta-Model (Collet et al., 1998; Vargas-Solar, 2000; Vargas-Solar & Collet, 2002) is defined in terms of dimensions associated to domains. Dimensions characterize different aspects of event processing phases: definition, detection, production, and notification. A domain gives the possible values for a dimension.

The event management model (i.e., an instance of the meta-model that restricts domains), supported by an event manager, specifies the possible protocols used to retrieve events, the methods to recognize (detect and order), compose, and notify events. In that way, an event type can be managed in different ways by the same manager according to consumer and producer needs.

Detection and production dimensions describe the conditions in which events are detected and how they are

recognized and ordered<sup>1</sup>. They include information to determine how component events contribute to composite events and how event parameters enter into the computation of the composite event parameters.

For an FDBMS, events can be produced within local or global transactions, ACID transactions, within distributed or non-classical transactions, or even within non-transactional applications. The validity of events (i.e., reliability of the information contained in events); and, in consequence, detection, production, and notification are determined by their context of production. Considering that events can stem from different transactional contexts and that they may even be invalidated by an abort of their producing transaction, event composition and notification policies should specify how to use event instances for event composition. When to notify them?

Event notification concerns information delivery, filtering, and lifespan of events. Notification takes place with respect to a particular event visibility. Consumers are notified about all instances of an event type (e.g., every instance of  $E_1$ ), or they may impose restrictions over the notification of the instances (e.g., all instances of  $E_1$  produced in Mexico). Moreover, events can be delivered to consumers at different instants, according to the context of production of events and on consumers needs.

Finally, it is necessary to establish interaction protocols depending on producer and consumer characteristics. Events are detected and notified under pull and push communication protocols. Both operations can be executed with an (a)synchronous mode. The synchronous (asynchronous) mode implies that producer (consumer) executions are (not) interrupted by the detection or the notification mechanism.

## RULE SERVICE

A rule service manages (define, update, activate, deactivate) and executes ECA active rules.

### Rule Definition

Defining a rule means to describe its ECA components (ON-IF-DO) and its behavior properties. Events describe FDBMS operations or external operations. Conditions are generally predicates on database states expressed with a query language. Actions are sequences of operations that can concern multiple DBMS, applications, the operating systems, and services. The behavior properties of the rule (i.e., CONSUMPTION, E\_PROCESSING, NET EFFECT, and COUPLING) depend on application needs, and they are defined according to the execution

model (see the next section). In our application example, we can define the rule  $R_{\text{dollar}}$  as follows:

```

RULE Rdollar
CONSUMPTION consume
E_PROCESSING instance
NET EFFECT off
COUPLING immediate, independent, separate

ON  after      (UpdateDollar, [9:00, 17:00])
    with      delta(price: real, country:
string)
        where country = 'France'

F   select    myaccount
    from      myaccount in Ac
counts
    where    myaccount.country =
"France" and
            myaccount.credit > 100000
DO  system(mail(myaccount. client name + '@bank.fr', 'Buy
dollars'));

```

The rule is triggered by events of type UpdateDollar that represent the dollar price in France produced after an update that occurred at an instant in [9:00, 17:00]. If the credit of the account myaccount in France is higher than 100000 dollars – which is tested by the Condition part of the rule – a mail is automatically sent to advise the owner to buy dollars. Here we assume that when evaluated, the query in the Condition will be rewritten according to the global catalogue. The following presents behavior properties determined by the execution model.

## Rule Execution

The core of the rule service is a parametric execution model based on a set of parameters and predefined values (Grazziotin-Ribeiro & Collet, 1999). This model specifies (1) when to execute a rule with respect to the notification of events and how to synchronize its execution with applications and FDBMS execution and (2) how to order possibly interrelated rules triggered at the same time.

Events are particularly important for rule execution, since they trigger rules. Event consumption policies specify how to handle an event that has triggered a rule. Events are considered for rule execution during their validity time interval. An event can be taken into account either for one execution of the rule or for several executions after its notification until the end of its validity time interval.

Rules can be triggered every time their triggering event is notified or only once for a set of triggering events. Furthermore, one can specify whether the net effect must be taken into account or not, for the execution of each rule. The net effect is the result of the execution of a sequence of operations on the same data (or object). In the previous example,  $R_{\text{dollar}}$  is triggered by events of type UpdateDollar.

Each event instance is considered for only one execution (CONSUMPTION consume). The rule is executed for each update of the dollar price in France (E\_PROCESSING instance-oriented). The net effect of update operations executed during the triggering transaction will not be computed (NET\_EFFECT off).

As said before, rule execution has to be coupled with the underlying transaction model of the FDBMS. Rule execution can start either immediately after the notification of its triggering event or it can be deferred, for example, to the end of the global triggering transaction<sup>2</sup>. A rule is executed either as a new transaction that belongs to the same global triggering transaction or as a separate global transaction. In addition, the global transaction in which a rule is executed can or cannot be dependent from the global triggering transaction. Clearly, coupling aspects imply that the rule service has both sufficient accesses over global transactions to be informed of their execution status and influence to abort, block, and restart them according to rule behavior properties. In our example, the execution of  $R_{\text{dollar}}$  starts immediately after the notification of events. The rule is executed as a separate independent global transaction (COUPLING immediate, independent, separate).

Finally, several rules can be triggered by the same event. When this situation comes up, one has to determine when and how these rules have to be executed. This is done by building an execution plan based on execution strategies specified according to different criteria (Collet, Vargas-Solar & Graziotin-Ribeiro, 2000; Graziotin-Ribeiro, 1998). In general, criteria combine sequential and parallel execution based upon rule dependencies. Sequential execution is achieved using rule ordering strategies that consider rule priorities, triggering and definition order, and execution cycles.

## RELATED WORKS

The research trend that aims to offer active functionalities as open services is recent. Basically, approaches intend to make active capabilities available for (non) database applications in distributed environments.

C<sup>2</sup>offein (Koschel et al., 1997) proposes separately usable components for distributed systems supported by CORBA. The system consists of parametric wrappers enabling event processing and a rule engine whose characteristics are tailored to specific applications. FRAMBOISE (Fritschi, Gatzu & Dittrich, 1998) proposes a construction system of ECA services decoupled from a particular DBMS. Event detection is performed by event detectors which have to be specialized for the respective DBMS. A rule service is responsible for the maintenance of the rule base and implements rule execution.



CoopWARE (Mylopoulos et al., 1996) and TriggerMan (Hanson & Khosla, 1997) also propose execution mechanisms for distributed rules. The former has been developed for a workflow environment, and the latter proposes an asynchronous trigger processor as an extension module for an object-relational DBMS. Chakravarthy et al. (1998) propose a framework for supporting ECA rules suitable for distributed and heterogeneous systems.

In general, these works resemble to ours insofar as they define active mechanisms to execute rules in distributed environments. Similar to us, rules can be triggered by events stemming from different sources. Event management in Koschel et al. (1997) and Frithschi et al. (1998) resembles ours as it supports different monitoring techniques with different qualities of communication protocols. However, none of them propose event managers that can be adapted dynamically at run-time. Different to us, in Koschel et al. (1997) and Hanson and Khosla (1997), rules are always executed as separate transactions, independent of the triggering transactions.

## CONCLUSION

Flexible infrastructures for supporting component-based distributed applications construction using rules are based on event and rule services used to specify, generate, and execute adaptable and extensible managers that may cooperate to detect events and execute rules in different contexts—that is, (non) transactional, centralized, distributed, and so forth.

Active services characteristics can be described by meta-models that specify policies for event type specification, event management, and reaction execution. A generation approach can be adopted to specify and implement open active database systems comprised of event and rule managers. In conclusion, active services for database federations can be implemented by services and managers as software components customizable according to (database) applications requirements and their environment (i.e., other services, the network, etc.).

Besides the generation approach that provides adaptability, open distributed active services (1) group together, in a flexible way (meta-modeling), functionalities proposed in non-classical (active) database domains and (2) use such solutions for improving communication and integration between database services and other components.

## REFERENCES

Bützingsloewen, G., Koschel, A., Lockemann, P., & Walter, H.D. (1998). ECA functionality in a distributed environment. In N.W. Paton (Ed.), *Active rules in database*

*systems, monographs in computer science* (pp. 147-175). Springer-Verlag.

Chakravarthy, S., Le, R., & Dasari, R. (1998). ECA rule processing in distributed and heterogeneous environments. *Proceedings of the 14th International Conference on Data Engineering*, Washington, February.

Collet, C., Vargas-Solar, G., & Grazziotin-Ribeiro, H. (1998). Toward a semantic event service for database applications. *Proceedings of the 9th International Conference, DEXA98* (pp. 16-27).

Collet, C., Vargas-Solar, G., & Grazziotin-Ribeiro, H. (2000). Active services for data-intensive distributed applications. *Proceedings of the International Symposium (IDEAS2000), IEEE*, 0-7695-0789-1 (pp. 349-359).

Coupaye, T. (1996). *Un modèle d'exécution paramétrique pour systèmes de bases de données actifs*. PhD thesis, University Joseph Fourier.

Coupaye, T., & Collet, C. (1998). Semantics based implementation of flexible execution models for active database systems. *Actes des 14ièmes Journées Bases de Données Avancées*, (pp. 405-424).

Fraternali, P., & Tanca, L. (1995, December). A structured approach for the definition of the semantics of active databases. *ACM Transactions on Database Systems*, 20(4), 414-471.

Fritschi, H., & Gatzju, S.K. (1998). FRAMBOISE: An approach to framework-based active database management system construction. *Proceedings of the 7th International Conference on Information and Knowledge Management*, ACM, 1-58113-061-9 (pp. 364-370).

Grazziotin-Ribeiro, H. (1999). *A rule service for federated database systems*. PhD thesis, University Joseph Fourier.

Grazziotin-Ribeiro, H., & Collet, C. (1999). Behavior of active rules within multi-database systems. *Proceedings of the XIV Symposium on Databases (SBBD)*, UFSC (pp. 55-69).

Hanson, E., & Khosla, S. (1997). *An introduction to the triggerman asynchronous trigger processor* (Tech. Rep. No. TR-97-007). University of Florida, CISE Department.

Koschel, A., Kramer, R., Bülzingslöwen, G., Bleibel, T., Krumlinde, P., Schmuck, S., & Wein, C. (1997). Configuration of active functionality for CORBA. *Proceedings of the ECOOP97 Workshop*.

Lamport L., & Melliar-Smit, P. (1985). Synchronizing clocks in the presence of faults. *Journal of the ACM*, 32(1), 52-78.

Mylopoulos, J., Gal, A., Kontogiannis, K., & Stanley, M. (1996). A generic integration architecture for cooperative information systems. *Proceedings of the 1st IFCIS International Conference on Cooperative Information Systems (CoopIS'96)* (pp. 208-217). IEEE.

Paton, N.W. (1998). *Active rules for databases*. Springer-Verlag.

Paton, N.W., Díaz, O., Williams, M.H., Campin, J., Dinn, A., & Jaime, A. (1993). Dimensions of active behaviour. *Proceedings of the 1st Workshop on Rules in Databases Systems (RIDS-93)*.

Schwidderki, S. (1996). *Monitoring the behaviour of distributed systems*. PhD thesis, University of Cambridge.

Vargas-Solar, G. (2000). *Service d'événements flexible pour l'intégration d'applications bases de données réparties*. PhD thesis, University Joseph Fourier.

Vargas-Solar, G., & Collet, C. (2002). ADEES, adaptable and extensible event service. *Proceedings of the 13th International Conference on Database Expert Systems and Applications (DEXA'02)*.

## KEY TERMS

**Active Mechanism:** A system responsible for detecting events and reacting automatically to such events according to predefined active rules or ECA rules. Traditionally, active mechanisms are embedded within Active Database Management Systems (ADBMS). Unbundled active mechanisms have been proposed for federated database systems and for component database systems.

**Active Rule:** It is represented by an ECA structure meaning *when an event is produced, if the condition is verified execute an action*. The event part represents a situation that triggers the rule, the condition part represents the state of a system execution or of a database, and the action part denotes a set of operations or a program.

**Event Instance:** An event is an instance of an event type associated to a point in time that belongs to the validity interval of its type.

**Event Management Model:** Defines policies for detecting, producing, and notifying instances of event types.

**Event Type:** Represents a class of situations produced within an event producer and that is interesting for a consumer. In the context of active systems, an event type represents the class of significant situations that trigger an active rule. Such situations are produced within a *validity interval* which is the time interval during which instances of the event type can be detected.

**Federated Database System:** A system that integrates a number of pre-existing autonomous DBMS which can be homogeneous or heterogeneous. They can use different underlying data models, data definition and manipulation facilities, and transaction management and concurrency control mechanisms. DBMS in the federation can be integrated by a mediator providing a unified view of data: a global schema, a global query language, a global catalogue and a global transaction manager. The underlying transaction model considers, in general, a set of transactions synchronized by a global transaction. Synchronization is achieved using protocols such as the Two-Phase Commit protocol.

**Rule Execution Model:** Defines policies for triggering a rule, evaluating its condition, and executing its action within the execution of an existing system (e.g., DBMS, FDBMS). It also defines policies used for executing a set of rules. Rule execution models have been characterized in taxonomies such as the one proposed by Coupaye, (1996); Paton et al. (1993); and Fraternali and Tanca (1995).

**Rule Model:** Describes the structure of a rule (i.e., its event, condition, and action parts).

## ENDNOTES

- <sup>1</sup> Events are ordered according to an instant of production that belongs to a global time that can be computed using methods as those proposed in Schwidderki (1996) and Lamport and Melliar-Smit (1985).
- <sup>2</sup> The triggering transaction is the transaction within which events are produced; a triggered transaction is the one that executes a triggered rule.

# Advanced Query Optimization

**Antonio Badia**

*University of Louisville, USA*

## INTRODUCTION

Query optimization has been an active area of research ever since the first relational systems were implemented. In the last few years, research in the area has experienced renewed impulse, thanks to new development like data warehousing. In this article, we overview some of the recent advances made in complex query optimization. This article assumes knowledge of SQL and relational algebra, as well as the basics of query processing; in particular, the user is assumed to understand cost optimization of basic SQL blocks (Select-Project-Join queries). After explaining the basic unnesting approach to provide some background, we overview three complementary techniques: source and algebraic transformations (in particular, moving outerjoins and pushing down aggregates), query rewrite (materialized views), new indexing techniques (bitmap and join indices), and different methods to build the answer (online aggregation and sampling). Throughout this article, we will use *subquery* to refer to a nested SQL query and *outer query* to refer to an SQL query that contains a nested query. The TPCB benchmark database (TPC, n.d.) is used as a source of examples. This database includes (in ascending size order) tables Nation, Customer (with a foreign key for Nation), Order (with a foreign key for Customer), and Lineitem (with a foreign key for Order). All attributes from a table are prefixed with the initial of the table's name ("c\_" for Customer, and so on).

## BACKGROUND

One of the most powerful features of SQL is its ability to express complex conditions using nested queries, which can be non-correlated or correlated. Traditional execution of such queries by the *tuple iteration* paradigm is known to be inefficient, especially for correlated subqueries. The seminal idea to improve this evaluation method was developed by Won Kim (1982) who showed that it is possible to transform many queries with nested subqueries in equivalent queries without subqueries. Kim divided subqueries into four different classes: non-correlated, aggregated subqueries (type-A); non-correlated, not aggregated subqueries (type-N); correlated, not aggregated subqueries (type-J); and cor-

related, aggregated subqueries (type-JA). Nothing can be done for type-A queries, at least in a traditional relational framework. Type-N queries correspond to those using IN, NOT IN, EXISTS, NOT EXISTS SQL predicates. Kim's observation was that queries that use IN can be rewritten transforming the IN predicate into a join (however, what is truly needed is a *semijoin*). Type-J queries are treated essentially as type-N.

Type-JA is the most interesting type. For one, it is a very common query in SQL. Moreover, other queries can be written as type-JA. For instance, a query with EXISTS can be rewritten as follows: a condition like EXISTS (SELECT attr...) (where attr is an attribute) is transformed into  $0 > (\text{SELECT COUNT}(*))$  (the change to "\*" is needed to deal with null values). To deal with type-JA queries, we first transform the subquery into a query with aggregation; the result is left in a temporary table, which is then joined with the main query. For instance, the query:

```
Select c_custkey
From Customer
Where c_acctbal > 10,000 and c_acctbal > (Select
sum(o_totalprice) From Order
      Where o_custkey = c_custkey))
```

is executed as:

```
Create Temp as
Select o_custkey, sum(o_totalprice) as sumprice
From Order
Group by o_custkey
```

```
Select c_custkey
From Customer, Temp
Where o_custkey = c_custkey and c_acctbal > 10,000
and c_acctbal > sumprice
```

However, this approach fails on several counts: first, non-matching customers (customers without Order) are lost in the rewriting (they will fail to qualify in the final join) although they would have made it into the original query (if their account balances were appropriate); this is sometimes called the *zero-count bug*. Second, the approach is incorrect when the correlation uses a predicate other than equality. To solve both problems at once, several authors suggested a new strategy: first, outerjoin the relations involved in the correlation (the



outerjoin will keep values with no match), then, compute the aggregation. Thus, the example above would be computed as follows:

```
Create Table Temp(ckey, sumprice) as
(Select c_custkey, sum(o_totalprice)
From Customer Left Outer Join Order on o_custkey =
c_custkey
Group by c_custkey)
```

```
Select c_custkey
From Customer, Temp
Where c_custkey = ckey and c_acctbal > 10,000 and
c_acctbal > sumprice
```

This approach still has two major drawbacks in terms of efficiency. First of all, we note that we are only interested in certain customers (those with an account balance over a given amount); however, all customers are considered in the outerjoin. The Magic Sets approach (Seshadri et al., 1996) takes care of this problem by computing first the values that we are interested in. For instance, in the example above, we proceed as follows:

```
Create table C_mag as
(Select c_custkey, c_acctbal From Customer
Where c_acctbal > 10,000);
```

```
Create table Magic as
(Select distinct c_custkey as key From C_mag);
/* this is the Magic Set */
```

```
Create table Correl (custkey, sumprice) as
(Select c_custkey, sum(o_totalprice)
From Customer outer join Magic on key =
c_custkey
Group by c_custkey)
```

```
Select c_custkey
from C_mag, Correl
where c_custkey = custkey and c_acctbal > sumprice
```

## QUERY OPTIMIZATION

### Query Transformations

Although the approaches discussed are an improvement over the naïve approach, they introduce another issue that must be taken care of: the introduction of outerjoins in the query plan is problematic, since outerjoins, unlike joins, do not commute among themselves and with other operators like (regular) joins and selections. This is a

problem because query optimizers work, in large part, by deciding in which order to carry out operations in a query, using the fact that traditional relational algebra operators can commute, therefore, can be executed in a variety of Order. To deal with this issue, Galindo-Legaria and Rosenthal (1997) give conditions under which an outerjoin can be optimized. This work is expanded in Rao et al. Intuitively, this line of work is based on two ideas: sometimes, outerjoins can be substituted by regular joins (for instance, if a selection is to be applied to the result of the outerjoin and the selection mentions any of the padded attributes, then all padded tuples that the outerjoin adds over a regular join will be eliminated anyway, since they contain null values and null values do not pass any condition, except the IS NULL predicate); and sometimes, outerjoins can be moved around (the generalized outerjoin proposed in the work above keeps some extra attributes so that interactions with joins are neutralized).

Finally, one last note on unnesting to point out that most approaches do not deal properly with operators involving negation or, equivalently, universal quantification, like operators involving the ALL comparison. It has been suggested that operators using ALL could be rewritten as antijoins; a query like:

```
Select c_custkey
From Customer
Where c_acctbal > ALL (Select o_totalprice From
Order where c_custkey = o_custkey)
```

can be decided by outerjoining Customer and Order on condition  $(c\_custkey = o\_custkey \text{ AND } c\_acctbal \leq o\_totalprice)$  (since a tuple in Customer would be present in the outerjoin only if the  $c\_acctbal$  was never less than or equal to a total price, that is, if the  $c\_acctbal$  was greater than all total prices); unfortunately, this reasoning does not hold when there are nulls present in either attributes. A different approach dealing with these operators is that of Aiken and Bahlen (2003). This work introduces a *multidimensional join* operator (MD), in which two relations can be joined on several conditions. The MD operator can be annotated with aggregate operations, each one computed in the result of a different join condition. Then, the above query would be computed by an MD-join between Customer and Order, where (a) a grouping by  $c\_custkey$  and a  $count(*)$  are computed over the join via condition  $c\_custkey = o\_custkey$ ; and (b) a grouping by  $c\_custkey$  and a  $count(*)$  are computed over the join via condition  $c\_custkey = o\_custkey$  and  $c\_acctbal > o\_totalprice$ . The tuples where the two counts are the same are then picked up by a selection. Intuitively, we count all tuples for a given value that could possibly fulfill the ALL operator and all tuples that actually fulfill it; if both numbers are the same, then

## Advanced Query Optimization

obviously all tuples have fulfilled the condition, and the value should be made part of the result. Another strategy to deal with this difficult case is outlined in Badia (2003), where a Boolean aggregate is proposed: after outerjoining Customer and Order and grouping by `c_custkey`, the aggregate simply ANDs comparisons between `c_acctbal` and all related `o_totalprice` values; if the final result is true, then the operator is fulfilled.

Another strategy developed recently in query optimization is the *pushing down of aggregates*. To push down an operator is simply to execute it earlier in the query process; it is well known that pushing down conditions is usually an effective way to reduce computation costs. For instance, in the query:

```
Select c_name, sum(c_acctbal)
From Customer, Nation
Where c_nationkey = n_nationkey and c_mktsegment =
"dairy"
Group by c_name
```

a result can be produced by joining Customer and Nation first and then selecting the appropriate customers, or by first selecting and then joining the result of the selection and the table Nation. In many cases, the latter is more effective, as it gets rid of undesired data sooner. However, the grouping (and the aggregate computation) must wait until all other operations are executed. To understand why it would be desirable to push down a group by. Note that the output of such an operation is usually (much) smaller than the input; like a selection, a group by tends to reduce the volume of data for further computation (there is another motivation, to be discussed later). Gupta, Harinayaran, and Quass (1995) study this issue and show that, in some cases, the group by can be pushed down; therefore, it can be done before joins or selections. Their basic idea, later extended in Goel and Iyer (1996), is that a group by breaks down a relation into groups; anything that treats such groups as a whole (instead of doing different things to different tuples in the group) is compatible with the grouping operation. Thus, in the case of selections, if a selection gets rid of all tuples in a group, or keeps all tuples in a group, the selection and the grouping can be done in any order. For instance, the query

```
Select c_name, sum(c_acctbal)
From Customer
Where c_name = "Jones"
Group by c_name
```

is one such case. The reason is that since all tuples in a group share the same name, they are all equally affected by the condition. What happens when this is not the case?

For instance, in our previous example, the condition clearly interferes with the grouping. However, we can first group by `c_name` and `c_mktsegment`; then apply the condition, and finally group by `c_name` alone. By adding the attribute in the selection condition to the grouping, we make groups that are not altered by selection. Note that now we have two groupings to do, but the second one is already computed by the first one. Note also that any sums computed by the first grouping can be “rolled up” to achieve the final, correct result.

As for the join, the problem can be attacked by looking at a join as a Cartesian product followed by a selection. We have already dealt with the issues introduced by selection; hence, we need to look at issues introduced by Cartesian products. The obvious issue is one of multiplicity. Assume the query:

```
Select c_name, sum(c_acctbal)
From Customer, Order
Where c_custkey = o_custkey
Group by c_name
```

Since all the attributes involved in the grouping and aggregation are from the Customer table, one may be tempted to push down the group to the Customer table before the join with Order. The problem with this approach is that it disturbs the computation of the aggregate; in effect, each tuple in Customer may match several tuples in Order; therefore, computing the sum before or after the join will yield different results. The basic observation here is the following: the Cartesian product may alter a computation by repeating some existing values but never by creating any new values (Chaudhuri & Shim, 1994). Some aggregates, like Min and Max, are *duplicate-insensitive*, that is, are not affected by the presence of duplicates; thus, there is nothing to worry about. Other aggregates like Sum, Count, and Avg are *duplicate-sensitive* (Chaudhuri & Shim, 1995). In this case, one needs to know how many duplicates of a value a join introduces in order to recreate the desired result. To do so, an aggregation is introduced on the other relation in the join, Order, with `count(*)` as the aggregate. This will give the desired information, which can then be used to calculate the appropriate final result. In our example above, we can group Customer by `c_name`, sum the `c_acctbal`, group Order by `o_custkey` and compute `count(*)` on each such group (that is, compute how many times a give customer key appears in Order); then both grouped relations can be joined and the original sum on the Customer side multiplied by the count obtained in the Order side. Note that the relations are grouped before the join, thereby probably reducing their sizes and making join processing much more efficient.

## Data Warehousing Optimizations

In a DW environment, it is common to have complex queries which involve joins of large tables and grouping of large tables. Therefore, some special indices and processing techniques have been developed to support these kinds of queries over large relations.

For joins, using *join indices* is a common strategy. A join index is a table containing rowids or tuple ids to tuples in several relations such that the join index represents (precomputes) the join of the relations. Thus, given relations R, S with attributes A, B respectively, the index join for the join of R and S on condition  $A=B$  is a binary table with entries of the form (Rtid, Stid), where Rtid is a pointer to a tuple  $t_R$  in R, Stid is a pointer to a tuple  $t_S$  in S such that  $t_R.A = t_S.B$ . Note that the pointer may be physical or logical; when it is logical, there is some kind of translation to a physical pointer so that we can, given a tid, locate a tuple on disk.

The same idea can be extended to more than two relations. In the context of a data warehouse, where most queries involve the join of the fact table and several dimension tables, a *star index* join can be constructed. A typical star index is one that contains rows of the form (Frowid, Drowid1..., Drowidn), with Frowid a pointer to the Fact table and each Drowidi a pointer to Dimension table i, indicating how the Fact table would join each dimension table row by row.

A problem with join indices, especially with star indices, is that they may grow very large. The star index described above has as many rows as the fact table, which is usually quite large. A possible improvement is the O'Neil-Graefe schema (Graefe & O'Neil, 1995), in which separate join indices are implemented for each dimension table (i.e., each join index reflects the join of the fact table and one of the dimension tables) using bitmaps; when a query requires several such indices to be used, the selected bitmaps are all ANDed together to do the join in one step.

Another index used in data warehousing is the *bitmap index*. A bitmap is a sequence of bits. A bitmap index is used to indicate the presence or absence of a value in a row. Assume relation R has  $m$  rows, and attribute A in R can have one of  $n$  values. For each value, we build a bitmap with  $m$  bits such that the  $i$ th bit is set to 1 if the given value of A is the one in the  $i$ th row of R and to 0 otherwise. For instance, attribute Rank with values Assistant, Associate, Full would generate 3 bitmaps; on each one of them, bit 35 would be set to 1 in the first bitmap if the value of Rank in record 35 is Assistant and to 0 otherwise. Note that the  $n$  bitmaps will be complementary: each bit will be 1 on only one of them. Therefore, the higher the value of  $n$ , the more sparse the bitmaps will be (i.e., the proportion of 0s to the total

number of bits will be high). Since this is wasteful, bitmaps are usually utilized when  $n$  is small. The value  $n$  is said to be the cardinality of the attribute; therefore, bitmaps are more useful with low cardinality attributes. Since less, more dense bitmaps can then be used. To make them useful for high cardinality attributes, bitmaps can be compressed, but this in turn presents other problems (see below). Another strategy used is to represent the values not by a single bit but by a collection of bits. For instance, if  $n = 8$ , we would need 8 bitmaps in the schema just introduced. Another approach is to devote 3 bits to represent each possible value of the attribute by one of the numbers 0..7 in binary (hence, a logarithmic number of bits is enough). Thus, such bitmaps are made up of small collections of bits.

The attraction of using bitmaps resides in two characteristics. First is the ability of most CPUs to process them in a very efficient manner. The second characteristic is that bitmaps have the advantage of providing access to the base relation in sorted order; therefore, sequential or list prefetch can be used. Note that in order to access data, we have to transform the number  $i$  for bits that are set to 1 to a pointer to the  $i$ th row of the base table; however, this usually can be achieved with low overhead. Furthermore, bit manipulation can be used for several problems, like range predicates, multiple predicates over one relation, or (in combination with join indices) for complex predicates involving joins and selections.

We can also use bitmap indices in the context of trees. In the leaf of a tree, we have a value and a list of Rowids pointing to the rows that have that value. When the list of Rowids is long, substituting it with a bitmap may reduce the size of the representation. IBM is said to have implemented trees that can decide to use list of Rowids or bitmaps on the fly, depending on the data.

Finally, another optimization that is proper of DW is the idea of using aggregate or summary tables (Gupta & Mumick, 1999). These are *materialized views* created by aggregating the fact table on some dimension(s). Whenever a query comes in that needs to aggregate the fact table on exactly those aggregates and dimensions (or sometimes, a subset of them, as some aggregates can be computed stagewise), we use the summary table instead of the fact table. Since the summary table is usually much smaller than the fact table due to the grouping, the query can be processed much faster. However, in order to use materialized views in a data warehouse, two questions must be answered:

- Many aggregations are possible, which ones to materialize? If a fact has  $n$ -dimensions, and each dimension has a hierarchy with  $m$  levels, in principle, there are  $m \times n$  ways to aggregate the data. In

fact, since any subset of values is a possible grouping, there are  $2^m \times n$  ways to summarize. Clearly, it is impossible to materialize all the views. Several issues have been researched: which views are more useful in answering queries (that is, which views would be used by more queries)? Which materialized aggregates will be a good trade-off of space (storage constraints) vs. time (query response time)? Given two views, can one be computed from the other so that it is redundant to materialize both of them? Most approaches are syntactical; that is, they do not take into account query workloads but try to determine a solution from the DW schema alone. A greedy algorithm, for example, can assign some utility and cost to each grouping, start with a low cost grouping, then expand on as far as the utility increases. These and other ideas are explored in the papers collected in Gupta and Mumick (1999).

- When can a query use a summary table instead of the original table? This is a complex question, since it may not be enough to check when the view is a subpart of the relational expression denoted by the query (so it can be substituted into it). For example, assume relation R and a view V defined by:

```
Select A, B, sum(D) From R Group By A, B
Assume that the query
Select A, B, C, sum(D) From R Group By A, B, C
```

is issued. This query can use view V since we can refine the grouping, and SUM can be computed stagewise. In order to deduce this, we can compare the query trees for V and the query; a subtree of the query tree can be made identical to the tree for V in part by pushing down the group by operation as described earlier (this is the other motivation for that line of work). But even query:

```
Select A, B, E, sum(D) From R, S Where R.A = S.Y
Group By A, B, E
```

with S.Y a primary key of S (so R.A is a foreign key) and E an attribute of S, can use V, although realizing so requires sophisticated reasoning. See Cohen, Nutt and Serebrenik (1999) for an approach to this issue.

## Approximate Answers

All approaches presented so far compute an exact answer, even if it takes a long time, and then return it. Lately, some work has investigated the idea of computing approximate answers with the goal of returning some useful information to the user in a much shorter

time. We sketch two ideas: sampling and online aggregation. Sampling is simply a development of statistical notions: if, instead of scanning a whole table, we scan a sample of it, we can produce a result very fast, even though it would not be exact. It is possible, however, to give a reasonable answer with a sample that is randomly obtained and has a minimum size. The trick in this approach is to obtain the right sample. If, for instance, we want to obtain a sample of the join of two large tables (say, Order and Lineitem), and we obtain a sample of Order and one of Lineitem and join them, the resulting sample may not be large enough because the space that it represents is that of the Cartesian product of Order and Lineitem. However, since this join Order has the primary key and Lineitem the foreign key, we can take a random sample of Lineitem (even a small one) and join the sample with Order (which can be quite efficient with a small sample). The result gives us a very good approximation of the join. A different approach is implemented in the Aqua system (Acharya, Gibbons & Poosala, 1999). Instead of using sampling, this system builds approximations, called *synopses*, on top of which an aggregate query can be run. For instance, the query:

```
Select sum(l_quantity)
from Order, Lineitem
where l_orderkey = o_orderkey and o_Ordertatus = F
```

is rewritten to

```
Select sum(l_quantity) * 100
from js_Order, bs_lineitem
where l_orderkey = o_orderkey and o_Ordertatus = F
```

where the tables js\_Order and bs\_lineitem are created by the system to implement a *join synopsis*, a 1% sample of the join between Order and Lineitem (that is why the resulting sum is scaled by 100).

Online aggregation focuses, like the name indicates, in queries involving aggregates. Because an aggregation produces one result from many data points, it is a primary target for computing approximate answers. The work at Berkeley (Haas & Hellerstein, 1999) is based on computing the aggregate incrementally and showing the results to the user, together with a confidence interval as they are computed. If the users decide they have enough accuracy, or enough information altogether, they can stop the computation. The basic sampling idea explained above is refined to form the basis of the *ripple join*, an algorithm designed to minimize the time to produce an initial, approximate answer with a reasonable level of precision (Haas & Hellerstein, 1999). This work is extended in Ioannidis and Poosala (1999). While previous work focused on aggregate queries and used



sampling-based techniques, this article presents a histogram-based approach, developing a histogram algebra that allows them to handle general queries. Since answers now can be sets, an error measure to quantify the error in an approximate query answer is also introduced. Finally, a related effort is that of Chaudhuri and Gravano (1999), in which queries requiring only the *top k* tuples of an answer (according to some ranking mechanism) can be efficiently evaluated using statistical information from the DBMS. An excellent, if a bit dated, tutorial on this area of research still ongoing is Haas and Hellerstein (2001).

## FUTURE TRENDS

Query optimization continues to be a very active and exciting area of research. Several areas seem to be currently favored by researchers: one is query optimization for XML with new techniques developed for this data model (tree matching, pattern and regular expression matching, tree indexing). Another one is the support for new types of answers: approximate, top-k, ranked answers can benefit from specific techniques. New indexing methods that use information in histograms and approximation techniques (especially for aggregate queries) are another important area. In the context of data warehousing, refinements to support cube operations and reuse of materialized views continue to be fundamental. Finally, support for operations on sets (like set joins, set containment) are also being studied, due to their wide applicability.

## CONCLUSION

Perhaps because of the practical interest of the subject, research on query optimization has never really stopped. There has been considerable advances in the area in the last years. Thanks in part to the development of data warehouses, new problems and techniques have been developed. In this article, we have overviewed some of the recent main ideas.

## REFERENCES

Acharya, S., Gibbons, P., & Poosala, V. (1999, September 7-10). Aqua: A fast decision support system using approximate query answers. *Proceedings of 25th International Conference on Very Large Data Bases*, Edinburgh, Scotland.

Akide, M., & Bahlen, M. (2003, March 5-8). Efficient computation of subqueries in complex OLAP. *Proceedings of the 19th International Conference on Data Engineering*, Bangalore, India.

Badia, A. (2003, September 3-5). Computing SQL subqueries with boolean aggregates. *Proceedings of the 5th International Data Warehousing and Knowledge Discovery Conference*, Prague, Czech Republic.

Chaudhuri, S., & Gravano, L. (1999, September 7-10). Evaluating top-k selection queries. *Proceedings of 25th International Conference on Very Large Data Bases*, September 7-10, 1999, Edinburgh, Scotland.

Chaudhuri, S., & Shim, K. (1994, September 12-15). Including group-by in query optimization. *Proceedings of the 20th International Conference on Very Large Data Bases*, Santiago de Chile, Chile.

Chaudhuri, S., & Shim, K. (1995). An overview of cost-based optimization of queries with aggregates. *Data Engineering Bulletin*, 18(3), 3-10.

Cohen, S., Nutt, W., & Serebrenik, A. (1999, June 14-15). Algorithms for rewriting aggregate queries using views. *Proceedings of the International Workshop on Design and Management of Data Warehouses*, Heidelberg, Germany.

Galindo-Legaria, C., & Rosenthal, A. (1997). Outerjoin simplification and reordering for query optimization. *ACM Transactions on Database Systems*, 22(1), 43-74.

Goel, P., & Iyer, B. (1996, June 4-6). SQL query optimization: Reordering for a general class of queries. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Montreal, Quebec, Canada.

Graefe, G., & O'Neil, P. (1995). Multi-table joins through bitmapped join indices. *SIGMOD Record*, 24(3), 8-11.

Gupta, A., Harinayaran, V., & Quass, D. (1995, September 11-15). Aggregate-query processing in data warehousing environments. *Proceedings of the 21st International Conference on Very Large Data Bases*, Zurich, Switzerland.

Gupta, A., & Mumick, I.S. (Eds.). (1999). *Materialized views: Techniques, implementations and applications*. Cambridge, MA: MIT Press.

Haas, P., & Hellerstein, J. (1999, June 1-3). Ripple joins for online aggregation. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Philadelphia, Pennsylvania.

Haas, P., & Hellerstein, J. (2001, May 21-24). Online query processing. Tutorial at the ACM SIGMOD International Conference on Management of Data Conference, Santa Barbara, California. Retrieved January 23, 2005, from <http://control.cs.berkeley.edu>

Ioannidis, Y., & Poosala, V. (1999, September 7-10). Histogram-based approximation of set-valued query-answers. *Proceedings of 25th International Conference on Very Large Data Bases*, Edinburgh, Scotland.

Kim, W. (1982). On optimizing an SQL-like nested query. *ACM Transactions on Database Systems*, 7(3), 443-469.

Rao, J., Lindsay, B., Lohman, G., Pirahesh, H., & Simmen, D. (2001, April 2-6). Using EELs, a practical approach to outerjoin and antijoin reordering. *Proceedings of the 17th International Conference on Data Engineering*, Heidelberg, Germany.

Seshadri, P., Hellerstein, J., Pirahesh, H., Leung, H.T., Ramakrishnan, R., Srivastava, D., et al. (1996, June 4-6). Cost-based optimization for Magic:Algebra and implementation. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Montreal, Quebec, Canada.

Seshadri, P., Pirahesh, H., & Leung, T.Y. (1996). Complex query decorrelation. *Proceedings of the 12th International Conference on Data Engineering*, February 26-March 1, New Orleans, Louisiana.

TPC. (n.d.). The TPCB benchmark. Retrieved January 23, 2005, from <http://www.tpc.org/tpch>

## KEY TERMS

**Bitmap Index:** An index containing a series of bitmaps such that for attribute A on relation R, each bitmap tells us if a given record in R has a certain value for A. Bitmap indices are often used in decision support environments, since used in conjunction with other bitmap or regular indices, they can cut down on disk accesses for selections, thereby improving query response time.

**Magic Set:** Given SQL query with a correlated, nested subquery, the magic set is the set of values that

are relevant for the computation of the subquery as parameters. This set is obtained by computing all conditions in the outer query except the one involving the subquery.

**Materialized View:** A view that is computed at definition time and stored on disk as a regular relation. Materialized views offer the advantage that can be used to precompute part of a query and then reused as many times as needed. The main challenge about materialized views is to keep them updated when the relations on which they were defined change; *incremental techniques* are used to make this efficient.

**Outerjoin:** Extension of the regular join operator. Given two relations R and S and a condition C involving attributes of R and S, the outerjoin will generate as its output: (a) all the tuples in the Cartesian product of R and S that respect condition C (i.e., a regular join); plus (b) all the tuples in R that do not have a match in S fulfilling C, padded with null values on the attributes of S; plus (c) all the tuples in S that do not have a match in R fulfilling C, padded with null values on the attributes of R. A left outerjoin will output only (a) and (b), and a right outerjoin will output only (a) and (c).

**Query Rewriting:** An approach to query optimization that rewrites the original SQL query into another equivalent SQL query (or a sequence of several SQL queries) that offer the possibility of yielding better performing query plans once they are processed by the query optimizer.

**Semijoin:** Variant of the regular join operator. The (left) semijoin of relations R and S on condition C involving attributes of R and S will generate as its output all the tuples in R that have at least one matching tuple in S that fulfills C (i.e., the tuples in R that would participate in a regular join) without repetitions (the right semijoin is defined similar). Algebraically, the (left) semijoin is defined as the projection on the schema of relation R of the join of R and S.

**Unnesting:** A particular technique in query rewriting that aims at transforming SQL queries with subqueries in them into equivalent queries without subqueries, also called *flattening*.

# Applying Database Techniques to the Semantic Web

**María del Mar Roldán-García**

*University of Malaga, Spain*

**Ismael Navas-Delgado**

*University of Malaga, Spain*

**José F. Aldana-Montes**

*University of Malaga, Spain*

## INTRODUCTION

Information on the Web has grown very quickly. The semantics of this information are becoming explicit and the Semantic Web (Berners-Lee, Hendler, & Lassila, 2001) is emerging. Ontologies provide a formal representation of the real world by defining concepts and relationships between them. In order to provide semantics to Web resources, instances of such concepts and relationships are used to annotate them. These annotations on the resources, which are based on ontologies, are the foundation of the Semantic Web. Because of the Web's size we have to deal with large amounts of knowledge. All this information must be represented and managed efficiently to guarantee the feasibility of the Semantic Web. Querying and reasoning over instances of ontologies will make the Semantic Web useful.

Knowledge representation is a well-known problem for artificial intelligence researchers. Explicit semantics is defined by means of formal languages. Description logics (Nardi & Brachman, 2003) is a family of logical formalisms for representing and reasoning on complex classes of individuals (called concepts) and their relationships (expressed by binary relations called roles). DL formalism allows the description of concepts, relationships, and individuals (i.e., the knowledge base). All of them, together with complex concept formation and concept retrieval and realization, provide a query/reasoning language for the knowledge base. Research in description logics deals with new ways to query a knowledge base efficiently. On the other hand, knowledge representation research does not deal with large amounts of information, and its results can only be applied to very small knowledge bases (with a small number of instances), which are not the knowledge bases we expect to find on the Semantic Web. As a consequence, reasoning algorithms are not scalable and usually are main-memory-oriented algorithms.

All these problems increase when we deal with the context of the Semantic Web. In such an environment,

reasoning and querying should be scalable, distributable, and efficient. In this context, we will find a fairly large amount of distributed instances. On the Semantic Web, not only reasoning on concepts is necessary, but also reasoning at the instance level and efficient instance retrieval. Therefore, it is necessary to allow (efficient) queries and (efficient) reasoning to be compatible in such a distributed (massive in instances) environment as the Semantic Web. This makes it necessary to develop efficient knowledge storage mechanisms that use secondary memory (in order to guarantee scale-up) and allow efficient reasoning about concepts and relationships defined by an ontology, and about its instances as well. Furthermore, it is fundamental to provide efficient disk-oriented reasoning mechanisms, particularly for the instance retrieval problem. We believe database technology (not necessarily database systems) is a must for this purpose, because it is developed to deal with large amounts of data and massive storage.

## BACKGROUND

### Description Logics

Description logics (DL) are a logical formalism, related to semantic networks and frame systems, for representing and reasoning on complex classes of individuals (called concepts) and their relationships (expressed by binary relations called roles). Typically, we distinguish between atomic (or primitive) concepts (and roles), and complex concepts defined by using DL constructors. Different DL languages vary in the set of constructors provided.

A DL knowledge base has two components: (1) a terminological part (the Tbox) that contains a set of concept descriptions and represents the general schema modeling the domain of interest and (2) an assertional part (the Abox) that is a partial instantiation of this schema,

Figure 1. Example of a description logic Tbox

String $\sqsubseteq$ Thing
Person $\sqsubseteq$ Thing
Professor $\sqsubseteq$ Person
Doctor $\sqsubseteq$ Professor
NoDoctor $\sqsubseteq$ Professor
Student $\sqsubseteq$ Person
PhDStudent $\sqsubseteq$ Student
UnderGradStudent $\sqsubseteq$ Student
Student $\sqsubseteq \forall \text{director.Professor}$
Student $\sqsubseteq \exists \text{name.String}$

comprising a set of assertions relating either individuals to classes or individuals to each other. Many of the applications only require reasoning in the Tbox, but in an environment like the Semantic Web, we also need Abox reasoning. Figure 1 shows an example of a description logic Tbox. Figure 2 shows an example of an Abox for this Tbox.

The reasoning tasks in a Tbox are consistency (satisfiability), which checks if knowledge is meaningful; subsumption, which checks whether all the individuals belonging to a concept (the subsumee) also belong to another concept (the subsumer); and equivalence, which checks if two classes denote the same set of instances (Nardi & Brachman, 2003). All of these reasoning mechanisms are reducible to satisfiability as long as we use a concept language closer under negation.

Typically, the basic reasoning tasks in an Abox are instance checking, which verifies whether a given individual is an instance of (belongs to) a specified concept; knowledge base consistency, which amounts to verifying whether every concept in the knowledge base admits at least one individual; and realization, which finds the most specific concept an individual object is an instance of (Nardi & Brachman, 2003).

Figure 3. OWL class constructors

Constructor Name	DL syntax
intersectionOf	$C_1 \sqcap \dots \sqcap C_n$
unionOf	$C_1 \sqcup \dots \sqcup C_n$
complementOf	$\neg C$
oneOf	$\{x_1 \dots x_n\}$
allValuesFrom	$\forall R.C$
someValuesFrom	$\exists R.C$
hasValue	$\exists R.\{x\}$
minCardinality	$\geq nR$
maxCardinality	$\leq nR$
cardinality	$= nR$

Figure 2. Example of an Abox for the Figure 1 Tbox

Roldan: Person
Navas: Person
Aldana: Person
Roldan: NoDoctor
Navas: NoDoctor
Aldana: Doctor
<Roldan, Aldana>: director
<Navas, Aldana>: director
<Roldan, "Maria del Mar">: name
<Navas, "Ismael">: name
<Aldana, "Jose F.">: name

In recent years significant advances have been made in the design of sound and complete algorithms for DLs. Moreover, systems using these algorithms have also been developed (Haarslev & Möller, 2001; Horrocks, 1998). Most of these approaches only deal with Tbox reasoning, but in an environment like the Semantic Web, we also need Abox reasoning. Although some DL systems provide sound and complete Abox reasoning, they provide very weak Abox query language. A query means retrieving instances that satisfy certain restrictions or qualifications and hence are of interest for a user.

## Web Ontology Languages

The recognition of the key role that ontologies play in the future of the Web has led to the extension of Web markup languages in order to provide languages to define Web-based ontologies. Examples of these languages are XML Schema,<sup>1</sup> RDF,<sup>2</sup> and RDF Schema (Brickley & Guha, 2002).

Even with RDF Schema, RDF has very weak semantics. Still, it provides a good foundation for interchanging data and enabling true Semantic Web languages to be layered

Figure 4. OWL axioms

Axiom	DL syntax
subClassOf	$C_1 \sqsubseteq C_2$
equivalentClass	$C_1 \equiv C_2$
subPropertyOf	$R \sqsubseteq S$
equivalentProperty	$R_1 \equiv R_2$
disjointWith	$C_1 \sqsubseteq \neg C_2$
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$
differentIndividual	$\{x_1\} \sqsubseteq \neg \{x_2\}$
AllDifferent	$\{x_1\} \sqsubseteq \neg \{x_1 \dots x_n\}$
inverseOf	$R_1 \equiv R_2^{-1}$
TransitiveProperty	$R^+ \sqsubseteq R$
SymmetricProperty	$R \equiv R^{-1}$
FunctionalProperty	$T \sqsubseteq \geq 1 R$
inverseFunctionalProperty	$T \sqsubseteq \geq 1 R^{-1}$



on top of it. By layering, we mean creating a language that uses the RDF syntax but also adds new classes and properties that have specific semantics.

A recent W3C working draft proposes OWL (Bechhofer et al., 2004) as an ontology language for the Semantic Web. From a formal point of view, OWL can be seen as an equivalent of a very expressive description logic, with an OWL ontology corresponding to a DL Tbox. Figures 3 and 4 show OWL constructors and OWL axioms and their equivalence with description logics. OWL is developed as a vocabulary extension of RDF Schema and is derived from DAML+OIL Web ontology language. This equivalence allows OWL to exploit the considerable existing body of description logics research.

OWL provides two sublanguages: OWL Lite for easy implementations and OWL DL to support the existing description logic business segment. OWL DL has desirable computational properties for reasoning systems. The complete language is called OWL Full.

## **DATABASE TECHNOLOGY FOR THE SEMANTIC WEB**

Description logic systems have been designed to deal with ontologies that are completely processable within a computer's primary memory. For real applications on the Semantic Web, we expect very large ontologies with very large amounts of instances. Therefore, current DL systems are not valid in this environment. Database technology is necessary in order to be able to deal with ontologies of this size.

Since RDF is the first XML syntax-based language to define ontologies, there are some approaches for storing RDF. These systems use a query language to query RDF schema and data information. ICS-FORTH RDFSuite (Alexaki, Christophides, Karvounarakis, Plexousakis, & Trolle, 2001) is a tool for RDF management. It has a tool for storing and querying RDF descriptions. This tool loads both the RDF schema and metadata defined according to the schema in an object-relational database. Furthermore, the tool distinguishes between unitary and binary relations holding the instances of classes and properties. Four tables are used to represent the RDF schemas, namely, Class, Property, SubClass, and SubProperty.

Sesame (Broekstra, Kampman, & Harmelen, 2002) is a system consisting of a repository, a query engine, and an administration module for adding and deleting RDF data and schema information. The system provides a generic API that allows using any storage device depending on the application domain. Two storage proposals in a relational database using PostgreSQL and MySQL are provided. They use the domain and range tables in addition to the

tables used by ICS-FORTH RDFSuite. The main difference between both proposals is that while in MySQL implementation of the database schema does not change when the RDF schema changes, PostgreSQL adds new tables to represent these changes. Both ICS-FORTH RDFSuite and Sesame use RQL (Karvounarakis, Alexaki, Christophides, Plexousakis, & Scholl, 2002) as the query language.

DLDB (Pan & Heflin, 2003) is a knowledge base that extends a relational database management system (RDBMS) with additional capabilities for DAML+OIL inference. Making use of the FaCT reasoner and Microsoft Access as a RDBMS, DLDB can process, store, and query DAML+OIL formatted semantic content. The system stores RDF in the relational database, and the reasoner is used to precompute the subsumption hierarchy. However, it does not support features that are not related to subsumption.

Logical databases provide a declarative representation language as well as efficient and persistent data storage. Groszof, Horrocks, Volz, and Decker (2003) propose a mapping of a DL subset into logic programs (LP) suitable for evaluation with Prolog. This intersection of DL with LP, called DLP, completely covers RDF Schema and a fraction of OWL. It allows "building ontologies on top of rules" (Groszof et al., 2003). An immediate result of this work is that LP engines could be used for reasoning with these ontologies and for reasoning with (possibly very large numbers of) facts. In Weithoener, Liebig, and Specht (2003), an alternative mapping, resulting in lower computational complexity and more representational flexibility, is presented. For the implementation, they have chosen the deductive database CORAL.

In Roldán and Aldana (2003), an approach using relational databases is presented. The knowledge representation, that is, ontologies, is expressed by the ALC DL (Schmidt-Schauß & Smolka, 1991). Queries are conjunctive queries defined for the ALC DL. A mapping between the ALC DL and the EER model is defined in such a way as to enable us to store our ontologies in the relational database in the most efficient manner. Two database schema levels are distinguished, because it is necessary not only to store the structure of the concrete knowledge base (called instance schema) but also the structure of any ALC knowledge base (called ontology schema) in order to store, for example, the IS-A hierarchy. On the other hand, a mapping between the query language (for conjunctive queries) and SQL is defined. Finally, the semantics of the functions that encapsulate the reasoning mechanisms must be defined. Once the functions that encapsulate the reasoning mechanisms are described, the access pattern defined for each of these functions must be analyzed in order to classify them in different

categories: (1) functions that have to be evaluated by the DL reasoner; (2) functions that have to be evaluated by the SQL reasoner; (3) functions that can be evaluated by both the DL and SQL reasoners; and (4) functions that have to be evaluated by both the DL and SQL reasoners.

## FUTURE TRENDS

The Semantic Web's aim is to be "machine understandable." This understanding is close to reasoning. We want to query classes and instances with respect to ontologies. We expect to find very large knowledge bases that entail reasoning with (a very large number of) instances. It is unlikely that standard Abox techniques will be able to cope with it. Therefore, we have to make reasoning scalable (both in the Tbox and in the Abox) in the context of the Semantic Web. It is very important that the Semantic Web become viable in a practical way. In order to achieve this, database technologies are fundamental; in fact, database technologies are a key issue to efficiently manage large amounts of knowledge.

Database technology is a natural branch of computer science that has demonstrated that it is able to cope with efficient and scalable queries over (possibly distributed) persistent data stored in a secondary memory.

In the Semantic Web context we are going to need all the knowledge developed by database research. This technology should be revised, adapted, or maybe redesigned, but we strongly believe that it is the only foundation that could lead to a Semantic Web that is usable in practice and beyond theoretical experiments.

In summary, persistence (structures for representation in secondary memory), access paths (indexes, maybe ad hoc indexing), and efficient and scalable queries (over data in mass storage and maybe distributed) are the successful technological issues that have to be adapted to the Semantic Web.

## CONCLUSION

The problem of combining the Semantic Web and database technology is not that simple nor that easy. We should still allow for reasoning in both the Tbox and (very specially) in the Abox. Furthermore, we have to combine queries and both types of reasoning in new query/reasoning languages.

There is still a great deal of work to be done, but this is the way to the future, not only for the Semantic Web but also for database technology.

To guarantee scalability we must merge artificial intelligence and database technology in the Semantic Web. It is necessary to define storage methods for knowledge,

that is, both representation methods and access paths (e.g., index). Furthermore, it is necessary to take into account that knowledge is not centralized in this context.

Nevertheless, we believe that there is not a unique representation and that a specific physical representation of the Abox and an ad hoc indexes structure will be necessary. Both representation and indexes will depend on the expressivity of the query language and the implemented reasoning. Therefore, the solution will not be unique, but a trade-off, and we will have hierarchies of ontology definition languages, each one with a representation trade-off (a best physical mapping) regarding both structural and query (generated by reasoning) patterns.

## REFERENCES

- Alexaki, S., Christophides, V., Karvounarakis, G., Plexousakis, D., & Trolle, K. (2001). The ICS-FORTH RDFSuite: Managing voluminous RDF description bases. *Secondnd International Workshop on the Semantic Web (SemWeb'01), CEUR Workshop Proceedings*, (Vol. 40).
- Berners-Lee, T., Hendler, J., & Lassila, O. (2001, May). The Semantic Web. *Scientific American*. Retrieved August 9, 2004, from <http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21>
- Brickley, D., & Guha, R. V. (2002). *RDF vocabulary description language 1.0: RDF Schema* (W3C Working Draft 30 April 2002). Retrieved August 9, 2004, from <http://www.w3.org/TR/rdf-schema/>
- Broekstra, J., Kampman, A., & Harmelen, F. (2002). Sesame: A generic architecture for storing and querying RDF and RDF Schema. *First International Semantic Web Conference (ISWC2002)*, Lecture Notes in Computer Science, (Vol. 2342).
- Grosov, B. N., Horrocks, I., Volz, R., & Decker, S. (2003). Description logic programs: Combining logic programs with description logic. In *Proceedings of the 12th International World Wide Web Conference*. New York: ACM Press.
- Haarslev, V., & Möller, R. (2001). RACER system description. In R. Goré, A. Leitsch, & T. Nipkow (Eds.), *Proceedings of International Joint Conference on Automated Reasoning, IJCAR'2001* (pp. 701-705). Berlin, Germany: Springer-Verlag.
- Horrocks, I. (1998). The FaCT system. In H. de Swart (Ed.), *Lecture notes in artificial intelligence: Vol. 1397. Automated reasoning with analytic tableaux and related*

*methods: International Conference Tableaux '98* (pp. 307-312). Berlin, Germany: Springer-Verlag.

Karvounarakis, G., Alexaki, S., Christophides, V., Plexousakis, D., & Scholl, M. (2002). RQL: A declarative query language for RDF. *WWW2002*, May 7-11, 2002, Honolulu, Hawaii.

Nardi, D., & Brachman, R. J. (2003). An introduction to description logics. In *The description logic handbook: Theory, implementation and applications* (pp. 1-40). Cambridge, UK: Cambridge University Press.

Bechhofer, S., Harmelen, F., Hendler, J., Horrocks, U. McGuinness, D., Patel-Schneider, P.F., & Stein, L.A. (2004). *OWL Web ontology language reference* (W3C Working Draft 10 February 2004). (2004). Retrieved August 9, 2004, from <http://www.w3.org/TR/owl-ref/>

Pan, Z., & Heflin, J. (2003). DLDB: Extending relational databases to support Semantic Web queries. In *Workshop on Practical and Scaleable Semantic Web Systems, ISWC 2003*, (pp. 109-113). Lehigh University: Dept. of Computer Science and Engineering.

Roldán, M. M., & Aldana J. F. (2003). Storing and querying knowledge in the deep Semantic Web. *Ière Conférence en Sciences et Techniques de l'Information et de la Communication (CoPSTIC'03)*. Morocco: Rabat.

Schmidt-Schauß, M., & Smolka, G. (1991). Attributive concept descriptions with complements. *Artificial Intelligence*, 48, 1-26.

Weithoener, T., Liebig, T., & Specht, G. (2003). Storing and querying ontologies in logic databases. In *First International Workshop on Semantic Web and Databases* (pp. 329-348). Berlin, Germany.

## KEY TERMS

**Assertional Reasoning:** A description logic knowledge base is made up of two parts, a terminological part (the terminology or Tbox) and an assertional part (the Abox), each part consisting of a set of axioms. The AbBox contains extensional knowledge that is specific to the individuals of the domain of discourse. There has been a great deal of work on the development of reasoning algorithms for expressive DLs, but in most cases these only consider Tbox reasoning. Reasoning mechanisms for the Abox (i.e., instance checking) isare called assertional reasoning. Assertional reasoning is important for real Semantic Web applications.

**Description Logics:** Description ILogics are considered the most important knowledge representation formalism unifying and giving a logical basis to the well-known traditions of fFrame-based systems, Ssemantic Nnetworks and KL-ONE-like languages, oObject-Oriented representations, Ssemantic data models, and Ttype systems.

**Knowledge Representation:** It is a fragmentary theory of intelligent reasoning, expressed in terms of three components: (1) the representation's fundamental conception of intelligent reasoning; (2) the set of inferences the representation sanctions; and (3) the set of inferences it recommends.

**Ontology:** An ontology is a logical theory accounting for the intended meaning of a formal vocabulary, i.e., its ontological commitment to a particular conceptualization of the world. The intended models of a logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualization) by approximating these intended models.

**Query/Reasoning:** A query means retrieving instances of both concepts and relationships between concepts, that satisfy certain restrictions or qualifications and hence are interesting for a user. Querying instances of ontologies involves making use of reasoning (both Abox and Tbox) mechanisms in order to obtain the corrected and completed results.

**Reasoning:** The ability of a system to find implicit consequences of its explicitly represented knowledge. Such systems are therefore characterized as knowledge-based systems.

**Semantic Web:** The Semantic Web is a mesh of information linked up in such a way as to be easily processable by machines, on a global scale. You can think of it as being an efficient way of representing data on the World Wide Web, or as a globally linked database.

## ENDNOTES

<sup>1</sup> <http://www.w3.org/XML/Schema/>

<sup>2</sup> <http://www.w3c.org/RDF/>

# Benchmarking and Data Generation in Moving Objects Databases

B

Theodoros Tzouramanis

University of the Aegean, Greece

## INTRODUCTION

Moving objects databases (MODs) provide the framework for the efficient storage and retrieval of the changing position of continuously moving objects. This includes the current and past locations of moving objects and the support of spatial queries that refer to historical location information and future projections as well. Nowadays, new spatiotemporal applications that require tracking and recording the trajectories of moving objects online are emerging. Digital battlefields, traffic supervision, mobile communication, navigation systems, and geographic information systems (GIS) are among these applications. Towards this goal, during recent years many efforts have focused on MOD formalism, data models, query languages, visualization, and access methods (Guting et al., 2000; Saltenis & Jensen, 2002; Sistla, Wolfson, Chamberlain, & Dao, 1997). However, little work has appeared on benchmarking.

## BACKGROUND

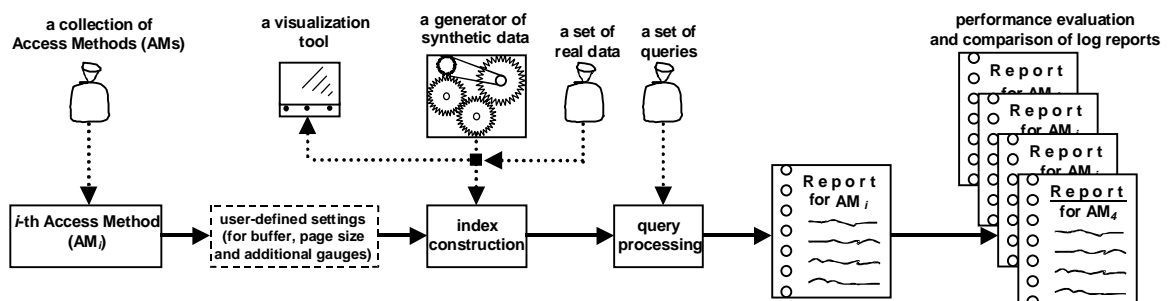
The role of benchmarking in MODs is to compare the behavior of different implementation alternatives under the same settings. It is important that the results must hold not only for a specific environment but in more general settings as well. Thus, the user is able to repeat the experiments and come to similar conclusions (Zobel, Moffat, & Ramamohanarao, 1996).

An example of a benchmark is the comparison of space requirements and query execution time of access methods for MODs. In order to compare the behavior of different indexing schemes under the same platform, there must be a flexible performance benchmarking environment. The general architecture of such a benchmarking toolkit is presented in Figure 1, and according to Theodoridis, Silva, and Nascimento (1999) and Tzouramanis, Vassilakopoulos, and Manolopoulos (2002), it consists of the following components:

- a collection of access methods for MODs,
- a module that generates synthetic data sets which cover a variety of real-life applications,
- a set of real data that also represents various real-life examples,
- a query processor capable of handling a large set of queries for extensive experimentation purposes,
- a reporter to collect all relevant output report logs, and
- a visualization tool to visualize data sets for illustrative purposes.

Good benchmarking must correspond to a recognizable, comprehensible real-life situation. In case large real data sets are not available, benchmarking requires the generation of artificial data sets following the real-world behavior of spatial objects that change their locations, shapes, and sizes over time and cover a variety of real-life applications. Thus one of the most important components of a benchmarking tool is the module that

Figure 1. A simplified benchmarking environment for access methods for MODs





generates synthetic data sets. In MODs, the work on the generation of synthetic data is limited, and only a few pioneering articles have recently addressed the topic of moving objects data generators.

## MAIN THRUST

### Synthetic Data Generators

A data set generator for MODs, called generator spatio-temporal data (GSTD), has been proposed by Theodoridis, Silva, and Nascimento (1999). It can generate moving points or rectangles and starts by distributing their centers in the workspace according to certain distributions. After the initialization phase, there are three main parameters to control the evolution of the objects throughout time, according to a desired distribution. These parameters are: (a) the duration of object instances, which involves time-stamp changes between consecutive instances; (b) the shift of the objects, which involves changes of spatial locations of the object centers; and (c) the resizing of objects, which involves changes of object sizes (only applicable to rectangular objects).

The GSTD supports three alternative approaches for the manipulation of invalid object instances in cases where an object leaves the spatial data space. However, a limitation of the GSTD approach is that the objects are moving almost freely in the workspace without taking into consideration the interaction between other objects or any potential restrictions. In particular, it could be argued that in any possible scenario, the objects are scattered all over the data space, or are moving in groups and the whole scene has the appearance of an unob-

structured polymorphic cloud movement. In the left part of Figure 2, a snapshot of an example of two synthetic data sets being displayed concurrently is illustrated.

In order to create more realistic scenarios, Pfooser and Theodoridis (2003) have extended the latter approach. They introduced an additional GSTD parameter to control the change of direction, and they used static rectangles for simulating an infrastructure, where the scenario indicates that each moving object has to be outside of these rectangles.

Saglio and Moreira (2001) propose the Oporto generator for time-evolving points and rectangles. It uses the modeling of fishing ships as a motivation. Ships are attracted by shoals of fish, while at the same time they are repulsed by storm areas. The fish themselves are attracted by plankton areas. Ships are moving points, whereas shoals, plankton, and storm areas are moving regions. Although useful for testing access methods, the original algorithm is highly specialized and turns out to be of limited use with respect to the demands of other real-world applications. In the right part of Figure 2, an example of the use of the Oporto generator is presented. The snapshot illustrates the motion of two ships being attracted by a gray shoal of fish.

Brinkhoff (2002) demonstrates a generator for “network-based” moving objects. It combines a real network with user-defined properties of the resulting data set. The driving application is the field of traffic telematics, and the presented generator satisfies exactly the requirements of this field. Important concepts of the generator are the maximum speed and the maximum edge capacity, the maximum speed of the object classes, the interaction between objects, the different approaches for determining the starting and the destination point of a moving object, and the recomputation of a route initiated by a

Figure 2. A snapshot of GSTD’s Web-based interface (left) and a snapshot of Oporto’s interface (right)

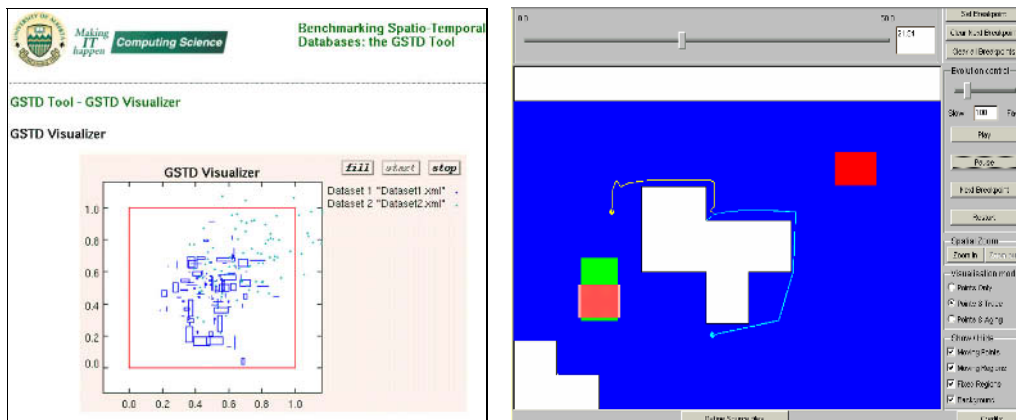
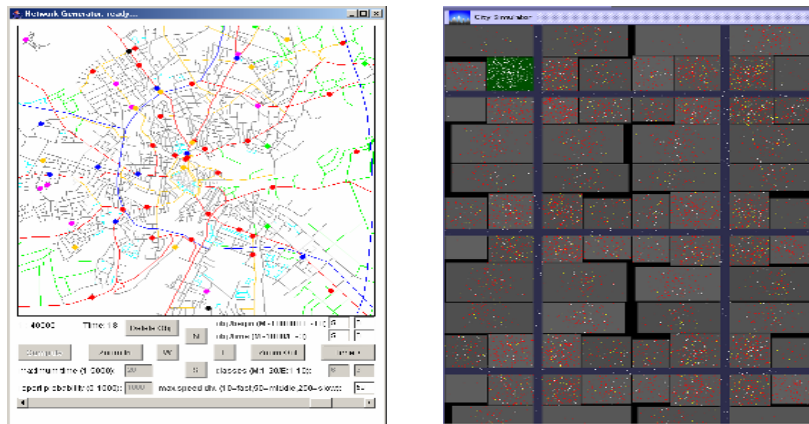


Figure 3. A snapshot of the network-based generator's interface (left) and a snapshot of the City Simulator's interface (right)

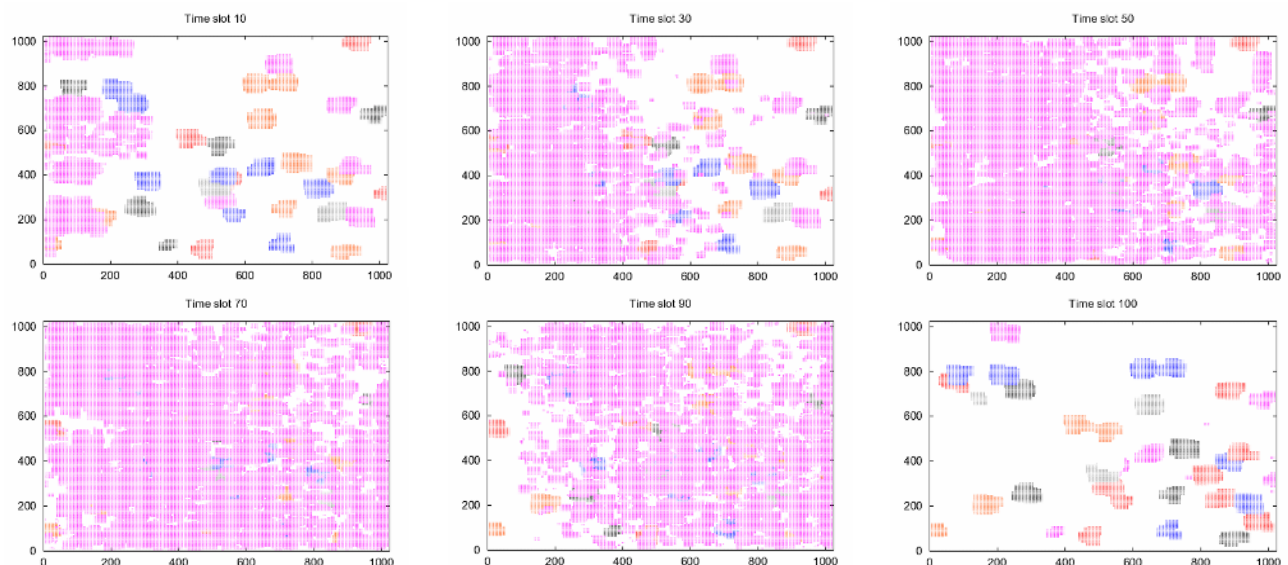


reduced speed on an edge and by external events. A framework for preparing the network, for defining functions and parameters by the users, and for data generation reporting are also presented. However, a disadvantage of the method is that it requires a significant computing power for the generation of moving objects. Nevertheless, although the author claims that the approach allows considering the characteristics of a wide range of applications (mostly network-based), it is clear that many other interesting real-world application examples such as meteorological phenomena, faunal phenomena, natural catastrophes, etc. cannot be simulated using this tool. In the left

part of Figure 3, an example of the use of the generator is illustrated. The points are moving objects and the lines are the roads.

The City Simulator (Kaufman, Myllymaki, & Jackson, 2001; Myllymaki & Kaufman, 2002) is another generator specifically designed for applications of the field of location-based services, i.e., for applications that must continuously track mobile users. It models the motion of large populations of people in a city. The data space of the city consists of roads, lawns, and buildings that affect the motion of the objects. Each building has an individual number of floors. Moving objects on a road

Figure 4. Snapshots of the G-TERD's interface



enter with a user-defined probability the first floor of a building. They perform random walks on this floor and may leave the building if they are near a door, or they may move up or move down to another floor, depending on user-defined probabilities, if they are near to the stairs. The number of objects, the number of time stamps, and many other parameters can be controlled by the user's interface. New city models can be imported as XML documents, which describe the layout of the city, including roads, buildings, and vacant space. In the right part of Figure 3 are shown the implemented city plan and an example of generated moving objects on the roads or on different floors of building.

Both the network-based generator and the City Simulator assume that the movement of objects is constrained to a transportation network. However, while the network-based generator is limited to two-dimensional data sets, the City Simulator supports the movement of objects in a three-dimensional workspace. Another difference is that in the case of the City Simulator, the movement of the objects is influenced by the surrounding place where they walk, while in the case of the network-based generator, their movement is influenced by other moving objects as well.

Finally, Tzouramanis et al. (2002) present another elaborate approach for synthetic data generation, which does not constrain movement to a network. It is called the generator of time-evolving regional data (G-TERD), and its basic concepts include the structure of complex two-dimensional regional objects, their color, maximum speed, zoom and rotation angle per time slot, the influence of other moving or static objects on the speed and on the moving direction of an object, the position and movement

of the scene observer, the statistical distribution of each changing factor, and, finally, the time.

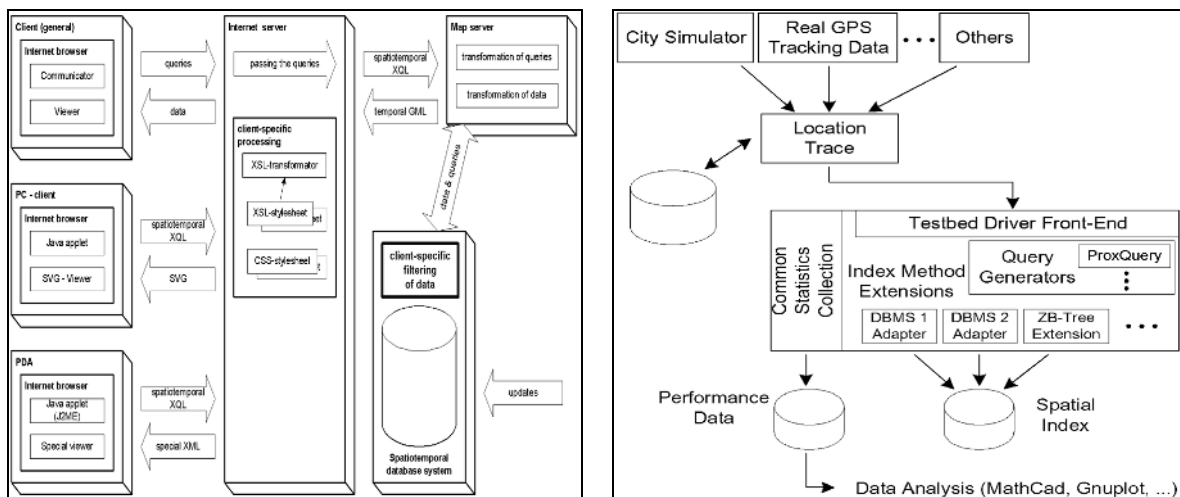
G-TERD is very parametric and flexible in the simulation of a variety of real-world scenarios. Users can specify the time domain and the workspace extent, the maximum number of objects in the sample, their minimum and maximum speed, zoom and rotation angle per time slot, the percentage of static and moving objects, the scene-observer's window size, and, finally, the number of colors supported. Users may also decide on the statistical model of the cardinality of the new objects and sub-objects per time slot, the deletion time slot of an object or sub-object, their speed, zoom, and rotation angle, and, finally, the time period that must elapse before the next computation of an attribute (speed, zoom, color of an object or sub-object, etc). Various statistical distributions are supported for that purpose. Figure 4 presents some snapshots of a realistic scenario produced by G-TERD. The snapshots show regional static objects that live for the whole scene lifetime and moving objects that are generated along the y-axis and cross the workspace but each at a different speed.

### Benchmarking Toolkits

In the sequel, we give a brief overview of two known benchmarking toolkits that have been designed for the evaluation of access methods for MODs (for more details, see Brinkhoff & Weitkamper, 2001; Myllymaki & Kaufman, 2002).

A benchmarking tool for querying XML-represented moving objects has been proposed by Brinkhoff and

Figure 5. The architectures of MOX (left) and LOCUS (right), as they have been proposed in Brinkhoff and Weitkamper (2001) and Myllymaki and Kaufman (2002), respectively



Weitkamper (2001). It is called MOX and its architecture is illustrated in the left part of Figure 5. An important requirement of applications using such an architecture is to be updated about new, reloaded, or removed objects fulfilling a given query condition. Such queries are called continuous queries. For testing the system and for investigating continuous queries, the network-based generator has been integrated into MOX. The objects produced by the generator are inserted into the database, which triggers the affected continuous queries.

LOCUS (Myllymaki & Kaufman, 2002) is another benchmarking environment for dynamic spatial indexing. The City Simulator is a component of its architecture for generating location trace files. These benchmark data are used as input for the construction of the indices of the access methods, the performance characteristics of which are tested by LOCUS. After the index construction,  $m$  spatiotemporal queries, such as proximity queries,  $k$ -nearest neighbor queries, and sorted-distance queries, are executed in respect to the location of a user. The architecture of LOCUS is depicted in the right part of Figure 5.

## FUTURE TRENDS

With the variety of spatiotemporal access methods for MODs, it becomes essential to have a framework to support the evaluation and fair comparison of these access methods. One approach is to use the already existing benchmark tools, i.e., the MOX and the LOCUS. Another approach is to develop a more general and extensive platform to capture the special needs of each class of the modeled application (e.g., the type of time-evolving spatial objects, the type of queries, and the frequency of updates). Much research work still needs to be carried out in this field in order to help the MOD research community to move in the right rigorous and experimental direction.

## CONCLUSION

This short survey has focused on the issue of benchmarking in MODs. For experimental evaluation of access methods and query processing techniques, real as well as synthetic data sets are important. The former is useful for ensuring that a technique under study is subject to realistic scenarios. For an interesting survey that covers several real data sets that are publicly available through the Web, the reader is referred to Nascimento, Pfoser, and Theodoridis (2003). However, the desired real data sets may not be available or they may

not be useful for testing extreme conditions. In contrast to the real data sets, synthetic data generators allow the generation of data sets with specific properties, thus making it possible to subject a technique to a wide variety of applications. We have briefly described five realistic data generators for MODs: the GSTD, the Oporto, the network-based generator, the City Simulator, and the G-TERD.

The network-based generator and the City Simulator have been integrated into the environment of the benchmarking toolkits MOX and LOCUS, respectively. In this process we have also described the general architecture of both these flexible tools, which have been specifically designed for the performance evaluation of access methods for MODs.

## REFERENCES

- Brinkhoff, T. (2002). A framework for generating network-based moving objects. *GeoInformatica*, 6(2), 155-182.
- Brinkhoff, T., & Weitkamper, J. (2001). Continuous queries within an architecture for querying XML-represented moving objects. In *Proceedings of the seventh International Symposium on Spatial and Temporal Databases* (Vol. 1, pp. 136-154). Berlin, Germany: Springer-Verlag.
- Guting, R. H., Bohlen, M. H., Erwig, M., Jensen, C. S., Lorentzos, N. A., Schneider, M., & Vazirgiannis, M. (2000). A foundation for representing and querying moving objects. *ACM Transactions on Database Systems*, 25(1), 1-42.
- Kaufman, J., Myllymaki, J., & Jackson, J. (2001). *City Simulator*. alphaWorks Emerging Technologies. New York: IBM Corporation.
- Myllymaki, J., & Kaufman, J. (2002). LOCUS: A testbed for dynamic spatial indexing. *Bulletin of the Technical Committee on Data Engineering*, 25(2), 48-55.
- Nascimento, M. A., Pfoser, D., & Theodoridis, Y. (2003). Synthetic and real spatiotemporal datasets. *Bulletin of the Technical Committee on Data Engineering*, 26(2), 26-32.
- Pfoser, D., & Theodoridis, Y. (2003). Generating semantics-based trajectories of moving objects. *International Journal of Computers, Environment and Urban Systems*, 27(3), 243-263.
- Saglio, J.-M., & Moreira, J. (2001). Oporto: A realistic scenario generator for moving objects. *GeoInformatica*, 5(1), 71-93.



Saltenis, S., & Jensen, C. S. (2002). Indexing of moving objects for location-based services. In *Proceedings of International Conference on Data Engineering* (pp. 463-472).

Sistla, A. P., Wolfson, O., Chamberlain, S., & Dao, S. (1997). Modeling and querying moving objects. In *Proceedings of International Conference on Data Engineering, IEEE Computer Society*, Los Alamitos, CA (pp. 422-432).

Theodoridis, Y., Silva, J. R. O., & Nascimento, M. A. (1999). On the generation of spatiotemporal datasets. In *Proceedings of the sixth Symposium on Spatial Databases* (pp. 147-164).

Tzouramanis, T., Vassilakopoulos, M., & Manolopoulos, Y. (2002). On the generation of time-evolving regional data. *GeoInformatica*, 6(3), 207-231.

Zobel, J., Moffat, A., & Ramamohanarao, K. (1996). Guidelines for presentation and comparison of indexing techniques. *ACM SIGMOD Record*, 25(3), 10-15.

## KEY TERMS

**Access Method:** A data structure that enables fast access over the records of a database file. Careful tuning or selection of the appropriate access method is very important in database performance.

**Benchmark Toolkit:** A tool that has been proposed for fair comparisons of different implementation alternatives, mostly of access methods. Its main components are: a synthetic data generator and/or some sets of real data, a query processor, and a set of access methods whose behavior has to be investigated. The output of a benchmark is a set of values describing the performance of each access method for a given data set and a set of queries which were executed iteratively by the query processor. Often these values describe separately the input/output time and CPU time needed for the index construction and for the computation of the queries.

**Location-Based Services:** Can be described as applications, which react according to a geographic trigger. A geographic trigger might be the input of a town name, zip code, or street into a Web page, the position of a mobile phone user, or the precise position of someone's car as s/he is driving home from the office. Using the knowledge of where someone is or where s/he

intends to go is the essence of location-based services. The arrival of high bandwidth mobile networks has highlighted the potential development of location-based services.

**Moving Objects Database:** A spatiotemporal database that represents the dynamic attributes of continuously moving objects, including their past, current, and anticipated future location information.

**Spatial Database:** A database that represents, stores, and manipulates static spatial data types, such as points, lines, surfaces, volumes, and hyper-volumes in multidimensional space.

**Spatiotemporal Database:** A database that manipulates spatial data, the geometry of which changes dynamically. It provides the chronological framework for the efficient storage and retrieval of all the states of a spatial database over time. This includes the current and past states and the support of spatial queries that refer to present and past time-points as well.

**Spatiotemporal Query:** A query that specifies spatial/temporal predicates and retrieves all objects that satisfy them. A spatial predicate is defined in terms of a point or an extent while a temporal predicate can involve a time instant or a time interval. For example, the query "Find all vehicles that will intersect the polygon S within the next five minutes" is a spatiotemporal range query. The spatial range is the polygon S and the temporal range is the time interval between now and 5 minutes from now.

**Synthetic Data Generator:** A tool able to generate a variety of synthetic data sets, according to statistical distributions. A fundamental issue in the generation of the synthetic data sets is the availability of a rich set of parameters that control their generation. Different statistical distributions and parameter settings correspond to different data sets and scenarios of real-world behavior.

**Synthetic Data Set:** A data set that is generated by some artificial specifications. Synthetic data are of high importance for the evaluation of the performance behavior of access methods in exactly specified or extreme situations, where real data are not available.

**Temporal Database:** A database that supports the maintenance of time-evolving data and the satisfaction of specialized queries that are related to three notions of time for these data: the past, the present, and the future.

# Bioinformatics Data Management and Data Mining

**Boris Galitsky**

*Birbeck College University of London, UK*

## INTRODUCTION

Bioinformatics is the science of storing, extracting, organizing, analyzing, interpreting, and utilizing information from biological sequences and molecules. The focus of bioinformatics is the application of computer technology to the management of biological information. Specifically, it is the science of developing computer databases and algorithms to facilitate and expedite biological research, particularly in genomics. It has been mainly stimulated by advances in DNA sequencing and *genome* mapping techniques (Adams, Fields & Venter, 1994). Genomics is the discipline that studies genes and their functions, including the functional study of genes, their resulting proteins, and the role played by the proteins in the biochemical processes, as well as the study of human genetics by comparisons with model organisms such as mice, fruit flies, and the bacterium *E. coli*

The Human Genome Project has resulted in rapidly growing databases of genetic sequences. Genome includes all the genetic material in the chromosomes (loci of genetic information) of a particular organism; its size is generally given as its total number of base pairs. New techniques are needed to analyze, manage, and discover sequence, structure, and functional patterns or models from these large sequence and structural databases. High performance data analysis algorithms are also becoming central to this task (Pevsner, 2000; Wilkins et al., 1997).

Bioinformatics provides opportunities for developing novel data analysis methods. Some of the grand challenges in bioinformatics include protein structure prediction, homology search, multiple alignment, and phylogeny construction (the evolutionary relationships among organisms; the patterns of lineage branching produced by the true evolutionary history of the organisms being considered). Proteins are defined as large molecules composed of one or more chains of amino acids in a specific order; the order is determined by the base sequence of nucleotides in the gene that codes for the protein. Proteins are required for the structure, function, and regulation of the body's cells, tissues, and organs; and each protein has unique functions. Examples

are hormones, enzymes, and antibodies. Amino acids are the building blocks of proteins; only about 20 amino acids are used to build the thousands of kinds of proteins needed by living cells. Nucleotides are the building blocks of DNA and RNA, consisting of a nitrogenous base, a five-carbon sugar, and a phosphate group. Together, the nucleotides form codons (groups of three nucleotides), which when strung together, form genes that link to form chromosomes.

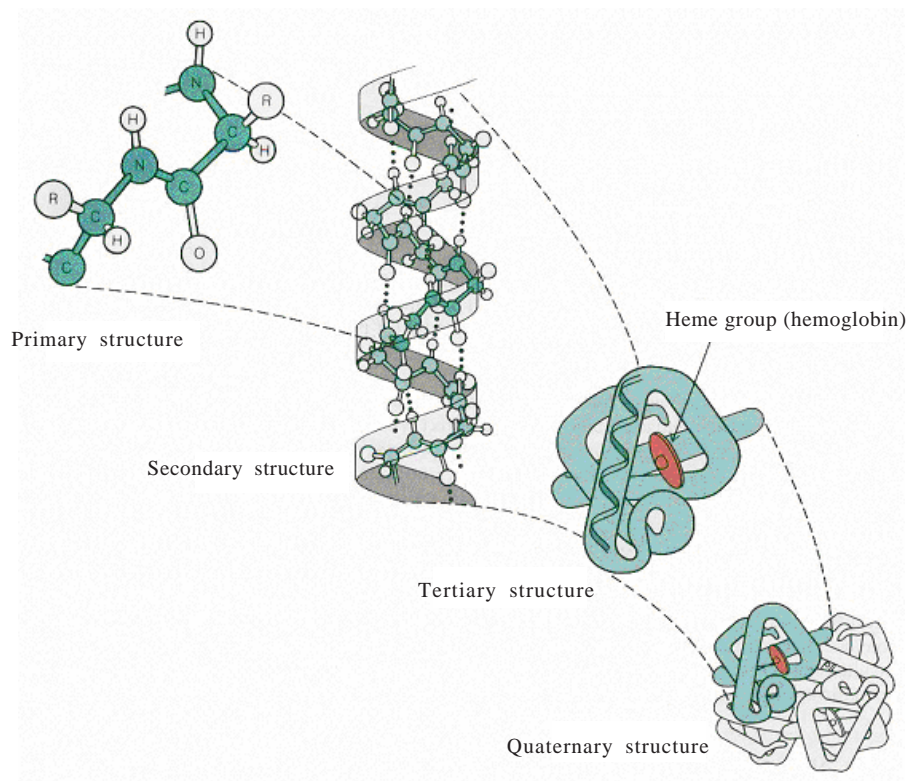
The other challenges of bioinformatics include genomic sequence analysis and gene finding, as well as applications in the data analysis of gene expression (transcriptional activity of a gene resulting in one or more RNA products and, usually following translation, one or more protein products), drug discovery in the pharmaceutical industry, and so forth (Waterman, 1995). For example, in protein structure prediction, the problem is in determining the secondary (division of sequence into fragments), tertiary (3D space), and quaternary structure of proteins, given their amino acid sequence (Figure 1) (Protein Structure).

Homology search aims at detecting increasingly distant homologues, that is, proteins related by evolution from a common ancestor. Multiple alignment and phylogenetic tree construction are interrelated problems. Multiple alignment aims at aligning a whole set of sequences to determine which subsequences are conserved. This works best when a phylogenetic tree of related proteins (illustration of how some of these proteins were inherited by others) is available. Finally, gene finding aims at locating the genes in a DNA sequence.

The following database-related problems are arising in bioinformatics:

- Data mining and warehousing as applied to biology;
- Data types and modeling needed for biological analysis;
- Interactive exploration and visualization for biology; and
- New indexing and search structures with applications to biology.

Figure 1. Illustration of the primary, secondary, tertiary, and quaternary structure of proteins



As to innovative approaches to database design, the following biological problems are worth mentioning:

1. Evolution and phylogenetic analysis (Maier et al., 2003). The demands of biodiversity and ecosystem research can advance one's understanding and use of information technologies;
2. Protein structure prediction;
3. Molecular sequence management and alignment;
4. Recognition of genes and regulatory elements;
5. Interpretation of large-scale gene expression data;
6. Whole genome comparative analysis and synthesis;
7. Modeling of biochemical pathways (complex network of interactions between proteins, Pathogenomics); and
8. Drug design and combinatorial libraries (Ghose & Viswanadha, 2001).

We also outline a number of challenges in representing biological data:

- The inherent complexity of biological data;
- Domain knowledge barrier;
- The evolution of domain knowledge; and
- The lack of expert data modeling skills.

## BACKGROUND

Data management for molecular and cell biology involves the traditional areas of data generation and acquisition, data modeling, data integration, and data analysis. In industry, the main focus of the past several years has been the development of methods and technologies supporting high-throughput data generation, especially for DNA sequence and gene expression data. New technology platforms for generating biological data present data management challenges arising from the need to: (1) capture, (2) organize, (3) interpret, and (4) archive vast amounts of experimental data. Platforms keep evolving with new versions benefiting from technological improvements, such as higher density arrays and better probe selection for microarrays.

This technology evolution raises the additional problem of collecting potentially incompatible data generated using different versions of the same platform, encountered both when these data need to be integrated and analyzed. Further challenges include qualifying the data generated using inherently imprecise tools and techniques and the high complexity of integrating data residing in diverse and poorly correlated repositories.

The data management challenges mentioned above, as well as other data management challenges, have been

examined in the context of both traditional and scientific database applications. When considering the associated problems, it is important to determine whether they require new or additional research, or can be addressed by adapting and/or applying existing data management tools and methods to the biological domain. The experts believe that existing data management tools and methods, such as commercial database management systems, data warehousing tools, and statistical methods, can be adapted effectively to the biological domain.

For example, the development of Gene Logic's gene expression data management system (GeneLogic) has involved modeling and analyzing microarray data in the context of gene annotations (including sequence data from a variety of sources), pathways, and sample annotations (e.g., morphology, demography, clinical), and has been carried out using or adapting existing tools. Dealing with data uncertainty or inconsistency in experimental data has required statistical, rather than data management, methods. (Adapting statistical methods to gene expression data analysis at various levels of granularity has been the subject of intense research and development in recent years.)

The most difficult problems have been encountered in the area of *data semantics*—properly qualifying data values (e.g., an expression estimated value) and their relationships, especially in the context of continuously changing platforms and evolving biological knowledge. While such problems are met across all data management areas, from data generation through data collection and integration to data analysis, the solutions require domain-specific knowledge and extensive work with data definition and structuring, with data management providing only the framework (e.g., controlled vocabularies, *ontologies*) to address these problems (The Gene Ontology Consortium, 2000; Gruber, 1993; Karp et al., 2000).

In an industry setting, solutions to data management challenges need to be considered in terms of complexity, cost, robustness, performance, and other user- and product-specific requirements. Devising effective solutions for biological data management problems requires thorough understanding of the biological application, the data management field, and the overall context in which the problems are considered (GeneLogic). Inadequate understanding of the biological application and of data management technology and practices seem to present more problems than the limitations of existing data management technology in supporting biological data-specific structures or queries.

## DATA MINING

As to the data mining in bioinformatics, it is an important source of important discoveries, based on the combination of advanced algorithm of classification, clustering, and pattern recognition and prediction with interactive visualization tools (Bertone & Gerstein, 2001, Galitsky, Gelfand & Kister, 1998). Data mining (knowledge discovery in databases) is the methodology to extract interesting (nontrivial, implicit, previously unknown, and potentially useful) information or patterns from data in large databases. The following methods are the basis for successful data mining applications:

- **Statistical algorithms:** Statistical analysis systems such as SAS and SPSS have been used by analysts to detect unusual patterns and explain patterns using statistical models such as linear models. Such systems have their place and will continue to be used.
- **Neural networks:** Artificial neural networks mimic the pattern-finding capacity of the human brain; hence, some researchers have suggested applying Neural Network algorithms to pattern-mapping. Neural networks have been applied successfully in a few applications such as secondary and tertiary structure prediction and microarray data analysis.
- **Genetic algorithms:** Optimization techniques that use processes such as genetic combination, mutation, and natural selection in a design based on the concepts of natural evolution.
- **Nearest neighbor method:** A technique that classifies each record in a dataset based on a combination of the classes of the *k* record(s) most similar to it in a historical dataset (sometimes called the *k*-nearest neighbor technique).
- **Rule induction:** The extraction of useful *if-then* rules from data based on statistical significance.
- **Data visualization:** The visual interpretation of complex relationships in multidimensional data. Has been applied in (Galitsky, 2003) for data mining of the sequences of immunoglobulin-like fold.

The comprehensive suite of bioinformatics data mining tools available, for example, at NCBI NIH (Baxevanis & Ouellette 1998; NIH Tools) includes the following:



- The *Basic Local Alignment Search Tool* (BLAST), for comparing gene and protein sequences against others in public databases.
- *Clusters of Orthologous Groups* (COGs) currently covers 21 complete genomes from 17 major phylogenetic lineages (the descendants of one individual). A COG is a cluster of very similar proteins found in at least three species. The presence or absence of a protein in different genomes can tell us about the evolution of the organisms, as well as point to new drug targets.
- *Map Viewer* shows integrated views of chromosome maps for 17 organisms. Used to view the NCBI assembly of complete genomes, including human, it is a valuable tool for the identification and localization of genes, particularly those that contribute to diseases.
- *LocusLink* combines descriptive and sequence information on genetic loci through a single query interface.
- A *UniGene Cluster* is a non-redundant (non-repetitive) set of sequences that represents a unique gene. Each cluster record also contains information such as the tissue types in which the gene has been expressed and map location.
- *Electronic PCR* (polymerase chain reaction: a method use to make multiple copies of DNA) allows you to search your DNA sequence for *sequence tagged sites*, which have been used as landmarks in various types of genomic maps.
- *VAST Search* is a structure–structure similarity search service. It compares tertiary structure (3D coordinates) of a newly determined protein structure to those in the PDB (Protein Data Bank) database. VAST Search computes a list of similar structures that can be browsed interactively, using molecular graphics to view superimpositions and alignments.
- The *Human–Mouse Homology Maps* compare genes in homologous segments of DNA from human and mouse sources, sorted by position in each genome.
- *Spidey* aligns one or more mRNA sequences to a single genomic sequence. Messenger RNA arises in the process of transcription from the DNA and includes information on the synthesis of a protein. *Spidey* will try to determine the exon/intron structure, returning one or more models of the genomic structure, including the genomic/mRNA alignments for each exon. Exon is a part of a gene that can encode amino acids in a protein. Usually adjacent to a non-coding DNA segment called an intron.

## FUTURE TRENDS

Genome database mining (Biodatabases.com) is referred to as computational genome annotation. Computational genome annotation is the identification of the protein-encoding regions of a genome and the assignment of functions to these genes on the basis of sequence similarity homologies against other genes of known function. Gene expression database mining is the identification of intrinsic patterns and relationships in transcriptional expression data generated by large-scale gene expression experiments. Proteome database mining is the identification of intrinsic patterns and relationships in translational expression data generated by large-scale proteomics experiments. As the determination of the DNA sequences comprising the human genome nears completion, the Human Genome Initiative is undergoing a paradigm shift from static structural genomics to dynamic functional genomics. Thus, gene expression and proteomics are emerging as the major intellectual challenges of database mining research in the postsequencing phase of the Human Genome Initiative.

Genome, gene expression, and proteome database mining are complementary emerging technologies with much scope being available for improvements in data analysis. Improvements in genome, gene expression, and proteome database mining algorithms will enable the prediction of protein function in the context of higher order processes such as the regulation of gene expression, metabolic pathways, and signaling cascades. The final objective of such higher-level functional analysis will be the elucidation of integrated mapping between genotype and phenotype (Bork et al., 1998).

## CONCLUSION

Bioinformatics may be alternatively defined as the interface between life sciences and computational sciences. It is a new science that has been stimulated by recent work on gene sequences; it applies the latest database techniques and smart mathematical algorithms to gene and protein sequence information in the search for new medical drug leads. Bioinformatics combines the storage and retrieval of complex biological data, with analysis and annotation of biological information. IT tools automate many of the processes, some of which take large amounts of computing power. The newest area is knowledge-based modeling of specific cellular and molecular processes. Bioinformatics is thus the study of the information content and information flow in biological systems and processes.

## REFERENCES

- Adams, M.D., Fields, C., & Venter J.C. (Eds.). (1994). *Automated DNA sequencing and analysis*. London: Academic Press.
- Baxeavanis, A., Ouellette, F.B.F. (Eds.). (1998). *Bioinformatics: A practical guide to the analysis of genes and proteins*. New York: John Wiley & Sons.
- Bertone, P., & Gerstein, M. (2001). Integrative data mining: The new direction in bioinformatics. *IEEE Engineering Medical Biology Magazine*, 20(4), 33-40.
- Biodatabases.com. Database mining tools in the Human Genome Initiative. Retrieved February 2, 2005, from <http://www.biodatabases.com/whitepaper03.html>
- Bork, P., Dandekar, T., Diaz-Lazcoz, Y., Eisenhaber, F., Huynen, M., & Yuan, Y. (1998). Predicting function: From genes to genomes and back. *Journal of Molecular Biology*, 283(4), 707-725.
- Galitsky, B. (2003). *Revealing the set of mutually correlated positions for the protein families of immunoglobulin fold*. In *Silico Biology 3*, 0022, Bioinformation Systems e.V. 241-264.
- Galitsky, B., Gelfand, I., & Kister, A. (1998). Predicting amino acid sequences of the antibody human V<sub>H</sub> chains from its first several residues. *Proceedings of the National Academy of Science*, 95 (pp. 5193-5198).
- The Gene Ontology Consortium. (2000). Gene ontology: Tool for the unification of biology. *Nature Genetics*, 25, 25-29.
- GeneLogic. Retrieved February 2, 2005, from <http://www.genelogic.com/>
- Ghose, A.K., & Viswanadha, V.N. (2001). *Combinatorial library design and evaluation: Principles, software tools, and applications in drug discovery*. New York: Marcel Dekker.
- Gruber, T.R. (1993). Towards principles for the design of ontologies used for knowledge sharing. *Proceedings of the International Workshop on Formal Ontology*.
- Karp, P.D., Riley, M., Saier, M., Paulsen, I.T., Paley, S.M., & Pellegrini-Toole, A. (2000). The EcoCyc and MetaCyc databases. *Nucleic Acids Research*, 28, 56-59.
- Maier, D., Landis, E., Cushing, J., Frondorf, A., Schnase, J.L., & Silberschatz, A. (2003). Information technology challenges of biodiversity and ecosystems informatics. *Information Systems*, 28, 241-242.

NIH Tools. Retrieved February 2, 2005, from <http://www.ncbi.nlm.nih.gov/Tools/>

Pathogenomics. Retrieved February 2, 2005, from <http://www.cmdr.ubc.ca/pathogenomics/terminology.html>

Pevsner, P. (2000). *Computational molecular biology: An algorithmic approach*. Cambridge, MA: MIT.

Protein Structure. Retrieved February 2, 2005, from <http://sosnick.uchicago.edu/precpsru.html> (1d-3d structure).

Waterman, M.S. (1995). *Introduction to computational biology: Maps, sequences, and genomes*. London: Chapman and Hall.

Wilkins, M.R., Williams, K.L., Appel, R.D., & Hochstrasser, D.H. (Eds.). (1997). *Proteome research: New frontiers in functional genomics*. Berlin: Springer-Verlag.

## KEY TERMS

**DNA:** Deoxyribonucleic acid. DNA molecules carry the genetic information necessary for the organization and functioning of most living cells and control the inheritance of characteristics

**Gene:** The unit of heredity. A gene contains hereditary information encoded in the form of DNA and is located at a specific position on a chromosome in a cell's nucleus. Genes determine many aspects of anatomy and physiology by controlling the production of proteins. Each individual has a unique sequence of genes, or genetic code.

**Genome:** It includes all the genetic material in the chromosomes of a particular organism; its size is generally given as its total number of base pairs.

**Messenger RNA:** Pieces of ribonucleic acid that carry genetic information from DNA to ribosomes (molecular machines that manufacture proteins), leading to their synthesis.

**Microarray:** Tool for studying how large numbers of genes interact with each other and how a cell's regulatory networks control vast batteries of genes simultaneously. Uses a robot to precisely apply tiny droplets containing functional DNA to glass slides. Researchers then attach fluorescent labels to DNA from the cell they are studying. The labeled probes are allowed to bind to cDNA strands on the slides. The slides are put into a scanning microscope to measure how much of a specific DNA fragment is present.



**Phylogenetic Tree:** A variety of dendrogram (diagram) in which organisms are shown arranged on branches that link them according to their relatedness and evolutionary descent.

**Phylogenetics:** The taxonomical classification of organisms based on their degree of evolutionary relatedness.

**Phylogeny:** Evolutionary relationships within and between taxonomic levels, particularly the patterns of lines of descent.

**Protein Structure Prediction:** The problem of determining the secondary (division of sequence into fragments), tertiary (3D space), and quaternary structure of proteins, given their amino acid sequence.

**RNA:** A single-stranded nucleic acid made up of nucleotides. RNA is involved in the transcription of genetic information; the information encoded in DNA is transcribed into messenger RNA (mRNA), which controls the synthesis of new proteins.

# Biological Data Mining

**George Tzanis**

*Aristotle University of Thessaloniki, Greece*

**Christos Berberidis**

*Aristotle University of Thessaloniki, Greece*

**Ioannis Vlahavas**

*Aristotle University of Thessaloniki, Greece*

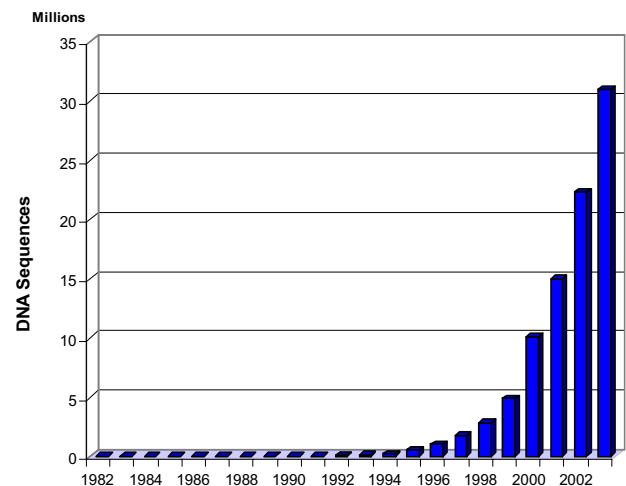
## INTRODUCTION

At the end of the 1980s, a new discipline named data mining emerged. The introduction of new technologies such as computers, satellites, new mass storage media, and many others have lead to an exponential growth of collected data. Traditional data analysis techniques often fail to process large amounts of, often noisy, data efficiently in an exploratory fashion. The scope of data mining is the knowledge extraction from large data amounts with the help of computers. It is an interdisciplinary area of research that has its roots in databases, machine learning, and statistics and has contributions from many other areas such as information retrieval, pattern recognition, visualization, parallel and distributed computing. There are many applications of data mining in the real world. Customer relationship management, fraud detection, market and industry characterization, stock management, medicine, pharmacology, and biology are some examples (Two Crows Corporation, 1999).

Recently, the collection of biological data has been increasing at explosive rates due to improvements of existing technologies and the introduction of new ones such as the microarrays. These technological advances have assisted the conduct of large-scale experiments and research programs. An important example is the Human Genome Project that was founded in 1990 by the U.S. Department of Energy and the U.S. National Institutes of Health (NIH) and was completed in 2003 (U.S. Department of Energy Office of Science, 2004). A representative example of the rapid biological data accumulation is the exponential growth of GenBank (Figure 1), the U.S. NIH genetic sequence database (National Center for Biotechnology Information, 2004). The explosive growth in the amount of biological data demands the use of computers for the organization, maintenance, and analysis of these data.

This led to the evolution of bioinformatics, an interdisciplinary field at the intersection of biology, computer science, and information technology. As Luscombe,

Figure 1. Growth of GenBank (years 1982-2003)



Greenbaum, and Gerstein (2001) mention, the aims of bioinformatics are:

- The organization of data in such a way that allows researchers to access existing information and to submit new entries as they are produced.
- The development of tools that help in the analysis of data.
- The use of these tools to analyze the individual systems in detail in order to gain new biological insights.

The field of bioinformatics has many applications in the modern day world, including molecular medicine, industry, agriculture, stock farming, and comparative studies (2can Bioinformatics, 2004).

## BACKGROUND

One of the basic characteristics of life is its diversity. Everyone can notice this by just observing the great

differences among living creatures. Despite this diversity, the molecular details underlying living organisms are almost universal. Every living organism depends on the activities of a complex family of molecules called proteins. Proteins are the main structural and functional units of an organism's cells. A typical example of proteins are the enzymes that catalyze (accelerate) chemical reactions. There are four levels of protein structural arrangement (conformation) as listed in Table 1 (Brazma et al., 2001). The statement about unity among organisms is strengthened by the observation that similar protein sets, having similar functions, are found in very different organisms (Hunter, 2004). Another common characteristic of all organisms is the presence of a second family of molecules, the nucleic acids. Their role is to carry the information that "codes" life. The force that created both the unity and the diversity of living things is evolution (Hunter, 2004).

Proteins and nucleic acids are both called biological macromolecules, due to their large size compared to other molecules. Important efforts towards understanding life are made by studying the structure and function of biological macromolecules. The branch of biology concerned in this study is called molecular biology.

Both proteins and nucleic acids are linear polymers of smaller molecules called monomers. The term *sequence* is used to refer to the order of monomers that constitute a macromolecule. A sequence can be represented as a string of different symbols, one for each monomer. There are 20 protein monomers called amino acids. There exist two nucleic acids, deoxyribonucleic acid (DNA) and ribonucleic acid (RNA), composed by four different monomers called nucleotides. DNA is the genetic material of almost every living organism. RNA has many functions

inside a cell and plays an important role in protein synthesis (Table 2). Moreover, RNA is the genetic material for some viruses such as HIV, which causes AIDS.

The genetic material of an organism is organized in long double-stranded DNA molecules called chromosomes. An organism may contain one or more chromosomes. Gene is a DNA sequence located in a particular chromosome and encodes the information for the synthesis of a protein or RNA molecule. All the genetic material of a particular organism constitutes its genome.

The central dogma of molecular biology, as coined by Francis Crick (1958), describes the flow of genetic information (Figure 2). DNA is transcribed into RNA, and then RNA is translated into proteins. The circular arrow around DNA denotes its replication ability. However, today it is known that in retroviruses RNA is reverse transcribed into DNA. Moreover, in some viruses, RNA is able to replicate itself. The extended statement of central dogma of molecular biology is depicted in Figure 3.

Houle et al. (2000) refer to a classification of three successive levels for the analysis of biological data that is identified on the basis of the central dogma of molecular biology:

1. Genomics is the study of an organism's genome and deals with the systematic use of genome information to provide new biological knowledge.
2. Gene expression analysis is the use of quantitative mRNA-level measurements of gene expression (the process by which a gene's coded information is converted into the structural and functional units of a cell) in order to characterize biological processes and elucidate the mechanisms of gene transcription (Houle et al., 2000).

Table 1. The four levels of protein conformation

<ul style="list-style-type: none"> <li>• <b>Primary Structure.</b> The sequence of amino acids, forming a chain called polypeptide.</li> <li>• <b>Secondary Structure.</b> The structure that forms a polypeptide after folding.</li> <li>• <b>Tertiary Structure.</b> The stable 3D structure that forms a polypeptide.</li> <li>• <b>Quaternary Structure.</b> The final 3D structure of the protein formed by the conjugation of two or more polypeptides.</li> </ul>
--

Table 2. Some of the basic types of RNA

<ul style="list-style-type: none"> <li>• <b>Messenger RNA (mRNA).</b> Carries information from DNA to protein synthesis site.</li> <li>• <b>Ribosomal RNA (rRNA).</b> The main constituent of ribosomes, the cellular components where the protein synthesis takes place.</li> <li>• <b>Transfer RNA (tRNA).</b> Transfers the amino acids to ribosomes.</li> </ul>
---

Figure 2. The central dogma of molecular biology (initial statement)



Figure 3. The central dogma of molecular biology (extended statement)



3. Proteomics is the large-scale study of proteins, particularly their structures and functions (Wikipedia, 2004).

These application domains are examined in the following paragraphs.

As many genome projects (the endeavors to sequence and map genomes) like the Human Genome Project have been completed, there is a paradigm shift from static structural genomics to dynamic functional genomics (Houle et al., 2000). The term structural genomics refers to the DNA sequence determination and mapping activities, while functional genomics refers to the assignment of functional information to known sequences. There are particular DNA sequences, that have a specific biological role. The identification of such sequences is a problem that concerns bioinformatics scientists. One such sequence is transcription start site, which is the region of DNA where transcription (the process of mRNA production from DNA) starts. Another biologically meaningful sequence is the translation initiation site, which is the site where translation (protein production from mRNA) initiates.

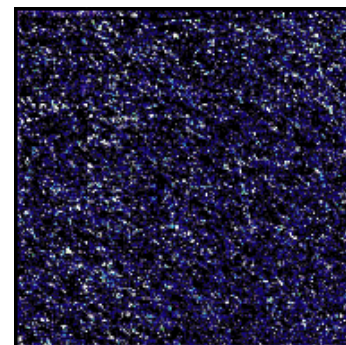
Although every cell in an organism—with only few exceptions—has the same set of chromosomes, two cells may have very different properties and functions. This is due to the differences in abundance of proteins. The abundance of a protein is partly determined by the levels of mRNA which in turn are determined by the expression or non-expression of the corresponding gene. A tool for analyzing gene expression is microarray. A microarray experiment measures the relative mRNA levels of typically thousands of genes, providing the ability to compare the expression levels of different biological samples. These samples may correlate with different time points taken during a biological process or with different tissue types

such as normal cells and cancer cells (Aas, 2001). An example raw microarray image is illustrated in Figure 4.

Serial Analysis of Gene Expression (SAGE) is a method that allows the quantitative profiling of a large number of transcripts (Velculescu et al., 1995). A transcript is a sequence of mRNA produced by transcription. However, this method is very expensive in contrast to microarrays; thus there is a limited amount of publicly available SAGE data.

One of the concerns of proteomics is the prediction of protein properties such as active sites, modification sites, localization, stability, globularity, shape, protein domains, secondary structure, and interactions (Whishart, 2002). Secondary structure prediction is one of the most important problems in proteomics. The interaction of proteins with other biomolecules is another important issue.

Figure 4. An illuminated microarray (enlarged). A typical dimension of such an array is about 1 inch or less, the spot diameter is of the order of 0.1 mm, for some microarray types can be even smaller (Brazma et al., 2001).



## MINING BIOLOGICAL DATA

Data mining is the discovery of useful knowledge from databases. It is the main step in the process known as Knowledge Discovery in Databases (KDD) (Fayyad et al., 1996), although the two terms are often used interchangeably. Other steps of the KDD process are the collection, selection, and transformation of the data and the visualization and evaluation of the extracted knowledge. Data mining employs algorithms and techniques from statistics, machine learning, artificial intelligence, databases and data warehousing, and so forth. Some of the most popular tasks are classification, clustering, association and sequence analysis, and regression. Depending on the nature of the data as well as the desired knowledge, there is a large number of algorithms for each task. All of these algorithms try to fit a model to the data (Dunham, 2002). Such a model can be either predictive or descriptive. A predictive model makes a prediction about data using known examples, while a descriptive model identifies patterns or relationships in data. Table 3 presents the most common data mining tasks (Dunham, 2002).

Many general data mining systems such as SAS Enterprise Miner, SPSS, S-Plus, IBM Intelligent Miner, Microsoft SQL Server 2000, SGI MineSet, and Inxight VizServer can be used for biological data mining. However, some biological data mining tools such as GeneSpring, SpotFire, VectorNTI, COMPASS, Statistics for Microarray Analysis, and Affymetrix Data Mining Tool have been developed (Han, 2002). Also, a large number of biological data mining tools is provided by the National Center for Biotechnology Information and by the European Bioinformatics Institute.

### Data Mining in Genomics

Many data mining techniques have been proposed to deal with the identification of specific DNA sequences. The most common include neural networks, Bayesian classifiers, decision trees, and Support Vector Machines (SVMs) (Hirsh & Noordewier, 1994; Ma & Wang, 1999; Zien et al., 2000). Sequence recognition algorithms exhibit perfor-

mance tradeoffs between increasing sensitivity (ability to detect true positives) and decreasing selectivity (ability to exclude false positives) (Houle et al., 2000). However, as Li, Ng, and Wong (2003) state, traditional data mining techniques cannot be directly applied to these types of recognition problems. Thus, there is the need to adapt the existing techniques to these kinds of problems. Attempts to overcome this problem have been made using feature generation and feature selection (Li et al., 2003; Zeng, Yap & Wong, 2002). Another data mining application in genomic level is the use of clustering algorithms to group structurally related DNA sequences.

### Gene Expression Data Mining

The main types of microarray data analysis include (Piatetsky-Shapiro & Tamayo, 2003) gene selection, clustering, and classification.

Piatetsky-Shapiro and Tamayo (2003) present one great challenge that data mining practitioners have to deal with. Microarray datasets—in contrast with other application domains—contain a small number of records (less than a hundred), while the number of fields (genes) is typically in the thousands. The same case is in SAGE data. This increases the likelihood of finding “false positives”.

An important issue in data analysis is feature selection. In gene expression analysis, the features are the genes. Gene selection is a process of finding the genes most strongly related to a particular class. One benefit provided by this process is the reduction of the foresaid dimensionality of dataset. Moreover, a large number of genes are irrelevant when classification is applied. The danger of overshadowing the contribution of relevant genes is reduced when gene selection is applied.

Clustering is the far most used method in gene expression analysis. Tibshirani et al. (1999) and Aas (2001) provide a classification of clustering methods in two categories: one-way clustering and two-way clustering. Methods of the first category are used to group either genes with similar behavior or samples with similar gene expressions. Two-way clustering methods are used to simultaneously cluster genes and samples. Hierarchical

Table 3. Common data mining tasks

Predictive	Descriptive
<p><b>Classification.</b> Maps data into predefined classes.</p> <p><b>Regression.</b> Maps data into a real valued prediction variable.</p>	<p><b>Association Analysis.</b> The production of rules that describe relationships among data.</p> <p><b>Sequence Analysis.</b> Same as association, but sequence of events is also considered.</p> <p><b>Clustering.</b> Groups similar input patterns together.</p>

clustering is currently the most frequently applied method in gene expression analysis. An important issue concerning the application of clustering methods in microarray data is the assessment of cluster quality. Many techniques such as bootstrap, repeated measurements, mixture model-based approaches, sub-sampling and others have been proposed to deal with the cluster reliability assessment (Kerr & Churchill, 2001; Ghosh & Chinnaiyan, 2002; Smolkin & Ghosh, 2003; Yeung, Medvedovic & Bumgarner, 2003).

In microarray analysis, classification is applied to discriminate diseases or to predict outcomes based on gene expression patterns and perhaps even to identify the best treatment for given genetic signature (Piatetsky-Shapiro & Tamayo, 2003).

Table 4 lists the most commonly used methods in microarray data analysis. Detailed descriptions of these methods can be found in literature (Aas, 2001; Dudoit, Fridly & Speed, 2002; Golub et al., 1999; Hastie et al., 2000; Lazzaroni & Owen, 2002; Tibshirani et al., 1999).

Most of the methods used to deal with microarray data analysis can be used for SAGE data analysis.

Finally, machine learning and data mining can be applied in order to design microarray experiments except to analyse them (Molla et al., 2004).

**Data Mining in Proteomics**

Many modification sites can be detected by simply scanning a database that contains known modification sites. However, in some cases, a simple database scan is not effective. The use of neural networks provides better results in these cases. Similar approaches are used for the prediction of active sites. Neural network approaches and nearest neighbor classifiers have been used to deal with protein localization prediction (Whishart, 2002). Neural networks have also been used to predict protein properties such as stability, globularity, and shape. Whishart refers to the use of hierarchical clustering algorithms for predicting protein domains.

Data mining has been applied for the protein secondary structure prediction. This problem has been studied for over than 30 years, and many techniques have been

developed (Whishart, 2002). Initially, statistical approaches were adopted to deal with this problem. Later, more accurate techniques based on information theory, Bayes theory, nearest neighbors, and neural networks were developed. Combined methods such as integrated multiple sequence alignments with neural network or nearest neighbor approaches improve prediction accuracy.

A density-based clustering algorithm (GDBSCAN) is presented by Sander et al. (1998) that can be used to deal with protein interactions. This algorithm is able to cluster point and spatial objects according to both their spatial and non-spatial attributes.

**FUTURE TRENDS**

Because of the special characteristics of biological data, the variety of new problems and the extremely high importance of bioinformatics research, a large number of critical issues is still open and demands active and collaborative research by the academia as well as the industry. Moreover, new technologies such as the microarrays led to a constantly increasing number of new questions on new data. Examples of hot problems in bioinformatics are the accurate prediction of protein structure and gene behavior analysis in microarrays. Bioinformatics demands and provides the opportunities for novel and improved data mining methods development. As Houle et al. (2000) mention, these improvements will enable the prediction of protein function in the context of higher order processes such as the regulation of gene expression, metabolic pathways (series of chemical reactions within a cell, catalyzed by enzymes), and signaling cascades (series of reactions which occur as a result of a single stimulus). The final objective of such analysis will be the illumination of the way conveying from genotype to phenotype.

**CONCLUSION**

The recent technological advances have led to an exponential growth of biological data. New questions on these

*Table 4. Popular microarray data mining methods*

One-way Clustering	Two-way Clustering	Classification
Hierarchical Clustering Self-organizing Maps (SOMs) K-means Singular Value Decomposition (SVD)	Block Clustering Gene Shaving Plaid Models	SVMs K-nearest Neighbors Classification/Decision Trees Voted Classification Weighted Gene Voting Bayesian Classification



data have been generated. Scientists often have to use exploratory methods instead of confirming already suspected hypotheses. Data mining is a research area that aims to provide the analysts with novel and efficient computational tools to overcome the obstacles and constraints posed by the traditional statistical methods. Feature selection, normalization, and standardization of the data, visualization of the results, and evaluation of the produced knowledge are equally important steps in the knowledge discovery process. The mission of bioinformatics as a new and critical research domain is to provide the tools and use them to extract accurate and reliable information in order to gain new biological insights.

## REFERENCES

- 2can Bioinformatics. (2004). Bioinformatics introduction. Retrieved February 2, 2005, from <http://www.ebi.ac.uk/2can/bioinformatics/index.html>
- Aas, K. (2001). *Microarray data mining: A survey*. NR Note, SAMBA, Norwegian Computing Center.
- Brazma, A., Parkinson, H., Schlitt, T., & Shojatalab, M. (2004). A quick introduction to elements of biology: Cells, molecules, genes, functional genomics, microarrays. Retrieved February 2, 2005, from [http://www.ebi.ac.uk/microarray/biology\\_intro.html](http://www.ebi.ac.uk/microarray/biology_intro.html)
- Dudoit, S., Fridlyand, J., & Speed, T.P. (2002). Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, 97(457), 77-87.
- Dunham, M.H. (2002). *Data mining: Introductory and advanced topics*. Upper Saddle River, NJ: Prentice Hall.
- Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., & Uthurusamy, R. (1996). *Advances in knowledge discovery and data mining*. Menlo Park, CA: AAAI Press/MIT Press.
- Ghosh, D., & Chinnaiyan, A.M. (2002). Mixture modeling of gene expression data from microarray experiments. *Bioinformatics*, 18, 275-286.
- Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P., et al. (1999). Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286(5439), 531-537.
- Han, J. (2002). How can data mining help bio-data analysis? *Proceedings of the 2nd ACM SIGKDD Workshop on Data Mining in Bioinformatics* (pp. 1-2).
- Hastie, T., Tibshirani, R., Eisen, M.B., Alizadeh, A., Levy, R., Staudt, L., et al. (2000). 'Gene shaving' as a method for identifying distinct sets of genes with similar expression patterns. *Genome Biology*, 1(2), research0003.
- Hirsh, H., & Noordewier, M. (1994). Using background knowledge to improve inductive learning of DNA sequences. *Proceedings of the 10th IEEE Conference on Artificial Intelligence for Applications* (pp. 351-357).
- Houle, J.L., Cadigan, W., Henry, S., Pinnamaneni, A., & Lundahl, S. (2004). Database mining in the human genome initiative. Whitepaper, Bio-databases.com, Amity Corporation. Retrieved February 2, 2005, from <http://www.biodatabases.com/whitepaper.html>
- Hunter, L. (2004). Life and its molecules: A brief introduction. *AI Magazine*, 25(1), 9-22.
- Kerr, M.K., & Churchill, G.A. (2001). Bootstrapping cluster analysis: Assessing the reliability of conclusions from microarray experiments. *Proceedings of the National Academy of Sciences*, 98 (pp. 8961-8965).
- Lazzeroni, L., & Owen, A. (2002). Plaid models for gene expression data. *Statistica Sinica*, 12(2002), 61-86.
- Li, J., Ng, K.-S., & Wong, L. (2003). Bioinformatics adventures in database research. *Proceedings of the 9th International Conference on Database Theory* (pp. 31-46).
- Luscombe, N.M., Greenbaum, D., & Gerstein, M. (2001). What is bioinformatics? A proposed definition and overview of the field. *Methods of Information in Medicine*, 40(4), 346-358.
- Ma, Q., & Wang, J.T.L. (1999). Biological data mining using Bayesian neural networks: A case study. *International Journal on Artificial Intelligence Tools, Special Issue on Biocomputing*, 8(4), 433-451.
- Molla, M., Waddell, M., Page, D., & Shavlik, J. (2004). Using machine learning to design and interpret gene-expression microarrays. *AI Magazine*, 25(1), 23-44.
- National Center for Biotechnology Information. (2004). Genbank statistics. Retrieved February 2, 2005, from <http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html>
- Piatetsky-Shapiro, G., & Tamayo, P. (2003). Microarray data mining: Facing the challenges. *SIGKDD Explorations*, 5(2), 1-5.
- Sander, J., Ester, M., Kriegel, P.-H., & Xu, X. (1998). Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery*, 2, (pp. 169-194).

Smolkin, M., & Ghosh, D. (2003). Cluster stability scores for microarray data in cancer studies. *BMC Bioinformatics*, 4, 36.

Tibshirani, R., Hastie, T., Eisen, M., Ross, D., Botstein, D., & Brown, P. (1999). *Clustering methods for the analysis of DNA microarray data* (Tech. Rep.). Stanford, CA: Stanford University.

Two Crows Corporation. (1999). *Introduction to data mining and knowledge discovery* (3rd ed.).

U.S. Department of Energy Office of Science. (2004). Human Genome Project information. Retrieved February 2, 2005, from [http://www.ornl.gov/sci/techresources/Human\\_Genome/home.shtml](http://www.ornl.gov/sci/techresources/Human_Genome/home.shtml)

Velculescu, V.E., Zhang, L., Vogelstein, B., & Kinzler, K.W. (1995). Serial analysis of gene expression. *Science*, 270(5235), 484-487.

Whishart, D.S. (2002). Tools for protein technologies. In C.W. Sensen (Ed.), *Biotechnology, (Vol 5b) genomics and bioinformatics* (pp. 325-344). Wiley-VCH.

Wikipedia Online Encyclopedia. (2004). Retrieved February 2, 2005, from <http://en2.wikipedia.org>

Yeung, Y.K., Medvedovic, M., & Bumgarner, R.E. (2003). Clustering gene-expression data with repeated measurements. *Genome Biology*, 4(5), R34.

Zeng, F., Yap C.H.R., & Wong, L. (2002). Using feature generation and feature selection for accurate prediction of translation initiation sites. *Genome Informatics*, 13, 192-200.

Zien, A., Ratsch, G., Mika, S., Scholkopf, B., Lengauer, T., & Muller, R.-K. (2000). Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics*, 16(9), 799-807.

## KEY TERMS

**Data Cleaning (Cleansing):** The act of detecting and removing errors and inconsistencies in data to improve its quality.

**Genotype:** The exact genetic makeup of an organism.

**Machine Learning:** An area of artificial intelligence, the goal of which is to build computer systems that can adapt and learn from their experience.

**Mapping:** The process of locating genes on a chromosome.

**Phenotype:** The physical appearance characteristics of an organism.

**Sequence Alignment:** The process to test for similarities between a sequence of an unknown target protein and a single (or a family of) known protein(s).

**Sequencing:** The process of determining the order of nucleotides in a DNA or RNA molecule or the order of amino acids in a protein.

**Visualization:** Graphical display of data and models facilitating the understanding and interpretation of the information contained in them.

# Biometric Databases

**Mayank Vatsa**

*Indian Institute of Technology, India*

**Richa Singh**

*Indian Institute of Technology, India*

**P. Gupta**

*Indian Institute of Technology, India*

**A.K. Kaushik**

*Ministry of Communication and Information Technology, India*

## INTRODUCTION

Biometric databases are the set of biometric features collected from a large public domain for the evaluation of biometric systems. For the evaluation of any algorithm for a particular biometric trait, the database should be a large collection of that particular biometric trait. The creation and maintenance of biometric databases involve various steps like enrollment and training.

Enrollment is storing the identity of an individual in the biometric system with his/her biometric templates. Every system has some enrollment policy, like the procedure for capturing the biometric samples, quality of the templates. These policies should be made such that they are acceptable to the public. For example, the procedure for capturing iris images involves the use of infrared light, which might even cause damage to human eyes. The signature of any individual is considered private for him, and he might have some objection to giving his signature for enrollment in any such system because of the unpredicted use of their templates (where and how this information may be used).

There are two types of enrollment, positive enrollment and negative enrollment. Positive enrollment is for candidates with correct identity civilians while negative enrollment is for unlawful or prohibited people.

For enrollment, the biometric sample captured should be such that the algorithms can perform well on that. The features which are required for any algorithm to verify or match the template should be present. The poor biometric sample quality increases the failure to enroll rate. For verification also, if the biometric sample is not captured properly this may lead to an increase in false acceptance and false rejection rates. This increase in failure to enroll and false acceptance and false rejection rates increases the manual intervention for every task, which leads to a further increase in the operational cost along with the risk in security.

Training is tuning the system for noise reduction, preprocessing, and extracting the region of interest so as to increase the verification accuracy (decreasing the false acceptance and false rejection rates).

## MAIN THRUST

There are various biometric databases available in the public domain for researchers to evaluate their algorithms. Given below are the various databases for different biometric traits along with their specifications and Web sites.

### Face Database

The face database is the necessary requirement for a face authentication system. The face database considers differing scaling factors, rotational angles, lighting conditions, background environment, and facial expressions.

Most of the algorithms have been developed using photo databases and still images that were created under a constant predefined environment. Since a controlled environment removes the need for normalization, reliable techniques have been developed to recognize faces with reasonable accuracy with the constraint that the database contains perfectly aligned and normalized face images. Thus the challenge for face recognition systems lies not in the recognition of the faces, but in normalizing all input face images to a standardized format that is compliant with the strict requirements of the face recognition module (Vatsa, Singh, & Gupta, 2003; Zhao, Chellappa, Rosenfeld, & Philips, 2000). Although there are models developed to describe faces, such as texture mapping with the Candide model, there are no clear definitions or mathematical models to specify what the important and necessary components are in describing a face.

## Biometric Databases

Using still images taken under restraint conditions as input, it is reasonable to omit the face normalization stage. However, when locating and segmenting a face in complex scenes under unconstrained environments, such as in a video scene, it is necessary to define and design a standardized face database.

There are several face databases available which can be used for face detection and recognition (see Table 1). The FERET database contains monochrome images taken in different frontal views and in left and right profiles. Only the upper torso of an individual (mostly head and neck) appears in an image on a uniform and uncluttered background. Turk and Pentland created a face database of 16 people. The face database from AT&T Cambridge Laboratories, formerly known as the Olivetti database and also known as the ORL-AT&T database, consists of 10 different images for 40 persons. The images were taken at different times, varying the lighting, facial expressions, and facial details. The Harvard database consists of cropped, masked frontal face images taken from a wide variety of light sources. There are images of 16 individuals in the Yale face database, which contains 10 frontal images per person, each with different facial expressions, with and without glasses, and under different lighting conditions. The XM2VTS multimodal database contains sequences of face images of 37 people. The five sequences for each person were taken over one week. Each image sequence contains images from right profile (-90 degree) to left profile (90 degree) while the subjects count from 0 to 9 in their native languages. The UMIST database consists of 564 images of 20 people with varying poses. The images of each subject cover a range of poses from right profile to frontal views. The Purdue AR database contains over 3,276 color images of 126 people (70 males and 56 females) in frontal view. This database is designed for face recognition experiments under several mixing factors such as facial expressions, illumination conditions, and occlusions. All the faces appear with different facial expressions (neutral, smile, anger, and scream), illumination (left light source, right light source, and sources from both sides), and occlusion (wearing sunglasses or scarf). The images were taken during two sessions separated by two weeks. All the images were taken by the same camera setup under tightly controlled conditions of illumination and pose. This face database has been applied to image and video indexing as well as retrieval. Vatsa et al. (2003) and Zhao et al. (2000) can be referred to for the details of preparation of face databases. These works have described how to prepare the face image database and the challenges in database preparation.

Table 1. Brief description of popular face databases

Face Database	Specifications	WWW Address
AR Face Database	4,000 color images of 126 people, with different facial expressions, illumination, and occlusions.	<a href="http://rvl1.ecn.purdue.edu/~aleix/aleix_face_DB.html">http://rvl1.ecn.purdue.edu/~aleix/aleix_face_DB.html</a>
AT&T Face Database	It consists of ten 92-by-112-pixel gray scale images of 40 subjects, with a variety of lighting and facial expressions.	<a href="http://www.uk.research.att.com/facedatabase.html">http://www.uk.research.att.com/facedatabase.html</a>
CVL Face Database	114 persons, 7 images for each person, under uniform illumination, no flash, and with projection screen in the background.	<a href="http://lrv.fri.uni-lj.si/facedb.html">http://lrv.fri.uni-lj.si/facedb.html</a>
CMU/VASC	2,000 image sequences from over 200 subjects.	<a href="http://vasc.ri.cmu.edu/idb/html/face/facial_expression/">http://vasc.ri.cmu.edu/idb/html/face/facial_expression/</a>
FERET Database	Largest collection of faces, over 14,000 images, with different facial expressions, illumination, and positions.	<a href="http://www.itl.nist.gov/iad/humanid/feret/">http://www.itl.nist.gov/iad/humanid/feret/</a>
Harvard Database	Cropped, masked frontal face images under a wide range of illumination conditions.	<a href="http://cvc.yale.edu/people/faculty/belhumeur.html">http://cvc.yale.edu/people/faculty/belhumeur.html</a>
IITK Face Database	1,000 color images of 50 individuals in lab environment, having different poses, occlusions, and head orientations.	<a href="http://www.cse.iitk.ac.in/users/mayankv">http://www.cse.iitk.ac.in/users/mayankv</a>
Extended M2VTS Database XM2VTSDB	XM2VTSDB contains four recordings of 295 subjects taken over a period of four months. Each recording contains a speaking head shot and a rotating head shot.	<a href="http://www.ee.surrey.ac.uk/Research/VSSP/xm2vtsdb">http://www.ee.surrey.ac.uk/Research/VSSP/xm2vtsdb</a>
M2VTS Multimodal Face Database	37 different faces and 5 shots for each person, taken at one-week intervals or when drastic face changes occurred, from 0 to 9 speaking head rotation and from 0 to 90 degrees and then back to 0 degrees, and once again without glasses if they wore.	<a href="http://www.tele.ucl.ac.be/PROJETS/CTS/M2VTS/">http://www.tele.ucl.ac.be/PROJETS/CTS/M2VTS/</a>
Max Planck Institute for Biological Cybernetics	7 views of 200 laser-scanned heads without hair.	<a href="http://faces.kyb.tuebingen.mpg.de/">http://faces.kyb.tuebingen.mpg.de/</a>
MIT Database	Faces of 16 people, 27 of each person, under various conditions of illumination, scale, and head orientation.	<a href="ftp://whitechapel.media.mit.edu/pub/images/">ftp://whitechapel.media.mit.edu/pub/images/</a>
NIST 18 Mug Shot Identification Database	Consists of 3,248 mug shot images, 131 cases with both profile and frontal views, 1,418 with only frontal view, scanning resolution is 500 dpi.	<a href="http://www.nist.gov/szrd/nistd18.htm">http://www.nist.gov/szrd/nistd18.htm</a>
UMIST Database	Consists of 564 images of 20 people, covering a range of different poses from profile to frontal views.	<a href="http://images.ee.umist.ac.uk/danny/database.html">http://images.ee.umist.ac.uk/danny/database.html</a>
University of Oulu Physics-Based Face Database	125 different faces, each in 16 different camera calibrations and illuminations, an additional 16 if the person has glasses, faces in frontal position captured under horizon, incandescent, fluorescent, and daylight illuminance.	<a href="http://www.ee.oulu.fi/research/imag/color/pbfd.html">http://www.ee.oulu.fi/research/imag/color/pbfd.html</a>
University of Bern Database	300 frontal-view face images of 30 people and 150 profile face images.	<a href="ftp://iamftp.unibe.ch/pub/Images/FaceImages/">ftp://iamftp.unibe.ch/pub/Images/FaceImages/</a>
Usenix	5,592 faces from 2,547 sites.	<a href="ftp://ftp.uu.net/published/usenix/faces/">ftp://ftp.uu.net/published/usenix/faces/</a>
Weizmann Institute of science	28 faces, includes variations due to changes in viewpoint, illumination, and three different expressions.	<a href="ftp.eris.weizmann.ac.il">ftp.eris.weizmann.ac.il</a> <a href="ftp.wisdom.weizmann.ac.il">ftp.wisdom.weizmann.ac.il</a>
Yale Face Database	165 images (15 individuals), with different lighting, expression, and occlusion configurations.	<a href="http://cvc.yale.edu">http://cvc.yale.edu</a>
Yale Face Database B	5,760 single light source images of 10 subjects, each seen under 576 viewing conditions (9 poses x 64 illumination conditions).	<a href="http://cvc.yale.edu/">http://cvc.yale.edu/</a>



## Fingerprint Database

Fingerprints are the ridge and furrow patterns on the tip of the finger and have been used extensively for personal identification of people. Fingerprint-based biometric systems offer positive identification with a very high degree of confidence. Thus, fingerprint-based authentication is becoming popular in a number of civilian and commercial applications. Fingerprints even have a number of disadvantages as compared to other biometrics. Some of the people do not have clear fingerprints; for example, people working in mines and construction sites get regular scratches on their fingers. Finger skin peels off due to weather, sometimes develops natural permanent creases, and even forms temporary creases when the hands are immersed in water for a long time. Such fingerprints can not be properly imaged with existing fingerprint sensors. Further, since fingerprints can not be captured without the user's knowledge, they are not suited for certain applications such as surveillance.

For the database, the user has the option of giving the fingerprint of any finger at the time of enrollment; of course, one should give the image of the same finger for correct matching at the time of verification also. The advantage with the generalization in considering the finger for authentication is that if anyone has a damaged thumb, then we can authenticate using the other finger. Also the performance and efficiency of any algorithm are absolutely independent of the choice of the finger. For the database, right-hand thumb images have been taken, but in case of any problem some other finger can also be considered. Some of the important issues of preparation of fingerprint image databases are discussed in the literature (Maltoni, Maio, Jain, & Prabhakar, 2003; Vatsa et al., 2003). The important issues are as follows.

1. **Resolution of image:** The resolution should be good enough to detect the basic features, i.e., loops, whorls, and arches in the fingerprint.
2. **The part of thumb in fingerprint image:** The fingerprint image must contain the basic features like whorls, arches, etc.
3. **The orientation of fingerprint:** The orientation is taken with the help of ridge flow direction. Because of the nature of fingerprint features, orientation also matters and thus is considered as a feature for recognition.

## Iris Database

Iris recognition has been an active research topic in recent years due to its high accuracy. The only database available in the public domain is CASIA Iris Database. To prepare

Table 2.

Fingerprint Database	Specifications	WWW Address
FVC 2000	4 databases, each database consists of fingerprints from 110 individuals and 8 images each. Images were taken at different resolutions by different scanners at 500 dpi. One database (DB4) contains synthetically generated fingerprints also.	<a href="http://bias.csr.unibo.it/fvc2000/databases.asp">http://bias.csr.unibo.it/fvc2000/databases.asp</a>
FVC 2002	4 databases, each consisting of 110 individuals with 8 images per individual at 500 dpi. One database (DB4) contains synthetically generated fingerprints also.	<a href="http://bias.csr.unibo.it/fvc2002/databases.asp">http://bias.csr.unibo.it/fvc2002/databases.asp</a>
NIST 4	Consists of 2,000 fingerprint pairs (400 image pairs from 5 classes) at 500 dpi.	<a href="http://www.nist.gov/srd/nistsd4.htm">http://www.nist.gov/srd/nistsd4.htm</a>
NIST 9	Consists of 5 volumes, each has 270 mated card pairs of fingerprints (total of 5,400 images). Each image is 832 by 768 at 500 dpi.	<a href="http://www.nist.gov/srd/nistsd9.htm">http://www.nist.gov/srd/nistsd9.htm</a>
NIST 10	Consists of fingerprint patterns that have a low natural frequency of occurrence and transitional fingerprint classes. There are 5,520 images of size 832 by 768 pixels at 500 dpi.	<a href="http://www.nist.gov/srd/nistsd10.htm">http://www.nist.gov/srd/nistsd10.htm</a>
NIST 14	27,000 fingerprint pairs with 832 by 768 pixels at 500 dpi. Images show a natural distribution of fingerprint classes.	<a href="http://www.nist.gov/srd/nistsd14.htm">http://www.nist.gov/srd/nistsd14.htm</a>
NIST 24	There are two components. First component consists of MPEG-2 compressed video of fingerprint images acquired with intentionally distorted impressions; each frame is 720 by 480 pixels. Second component deals with rotation; it consists of sequences of all 10 fingers of 10 subjects.	<a href="http://www.nist.gov/srd/nistsd24.htm">http://www.nist.gov/srd/nistsd24.htm</a>
NIST 27	Consists of 258 latent fingerprints from crime scenes and their matching rolled and latent fingerprint mates with the minutiae marked on both the latent and rolled fingerprints.	<a href="http://www.nist.gov/srd/nistsd27.htm">http://www.nist.gov/srd/nistsd27.htm</a>
NIST 29	Consists of 4,320 rolled fingers from 216 individuals (all 10 fingers) scanned at 500 dpi and scanned using WSQ. Two impressions of each finger are available in addition to the plain finger images.	<a href="http://www.nist.gov/srd/nistsd29.htm">http://www.nist.gov/srd/nistsd29.htm</a>
NIST 30	Consists of 36 ten-print paired cards with both the rolled and plain images scanned at 500 dpi and 1,500 dpi.	<a href="http://www.nist.gov/srd/nistsd30.htm">http://www.nist.gov/srd/nistsd30.htm</a>

Table 3.

Iris Database	Specification	WWW Address
CASIA Database	It includes 756 iris images from 108 classes. For each eye, 7 images are captured in two sessions, where 3 samples are collected in the first session and 4 in the second session.	<a href="http://www.sinobiometrics.com/resources.htm">http://www.sinobiometrics.com/resources.htm</a>

a database of iris images, it is required to have a setup consisting of a camera and lighting system. The details of the setup can be found in Daugman (1998) and Wildes (1999).

## Speaker Recognition Databases

Speaker recognition is finding the peculiar characters that the speaker possesses and reveals in his speech. The issues involved in designing a database for speaker recognition are the type of speech (text dependent or text independent), duration of the acquired speech, and acoustic environment (Campbell & Reynolds, 1999).

## Biometric Databases

Table 4.

Speaker Recognition Database	Specifications	WWW Address
NIST	Consists of 2,728 five-minute conversations with 640 speakers. For every call by the same person, different handsets were used; whereas, the receiver always had the same handset.	<a href="http://www.nist.gov/speech/tests/spk/index.htm">http://www.nist.gov/speech/tests/spk/index.htm</a>
TIMIT	It consists of 630 speakers, where the subjects read 10 sentences each.	<a href="http://www ldc.upenn.edu/Catalog/LDC93S1.html">http://www ldc.upenn.edu/Catalog/LDC93S1.html</a>
NTIMIT	The original TIMIT recordings were transmitted over long-distance phone lines and redigitized.	<a href="http://www ldc.upenn.edu/Catalog/LDC93S2.html">http://www ldc.upenn.edu/Catalog/LDC93S2.html</a>
YOHO Database	It consists of 128 subjects speaking the prompted digit phrases into high quality handsets. Four enrollment sessions were held, followed by 10 verification sessions at an interval of several days.	<a href="http://www ldc.upenn.edu/Catalog/LDC94S16.html">http://www ldc.upenn.edu/Catalog/LDC94S16.html</a>
OGI Database	It consists of telephone speech from about 1,084 participants. Twelve sessions were conducted over a 2-year period, with different noise conditions and using different handsets and telephone environments.	<a href="http://cslu.cse.ogi.edu/corpora/spkrec/">http://cslu.cse.ogi.edu/corpora/spkrec/</a>
ELRA—SIVA Database	It is an Italian speaker database consisting of 2,000 speakers. The recordings were made in a house or office environment over several sessions. It contains the data of imposters also.	<a href="http://www.elda.fr/catalogue/en/speech/S0028.html">http://www.elda.fr/catalogue/en/speech/S0028.html</a>
ELRA—PolyVar Database	The speakers are comprised of native and non-native speakers of French (mainly from Switzerland). It consists of 143 Swiss-German and French speakers with 160 hours of speech.	<a href="http://www.elda.fr/catalogue/en/speech/S0046.html">http://www.elda.fr/catalogue/en/speech/S0046.html</a>
ELRA—PolyCost Database	The database contains 1,285 calls (around 10 sessions per speaker) recorded by 133 speakers. Data was collected over ISDN phone lines with foreigners speaking English.	<a href="http://www.elda.fr/catalogue/en/speech/S0042.html">http://www.elda.fr/catalogue/en/speech/S0042.html</a>

Table 5.

Online Handwriting Database	Specifications	WWW Address
UPINEN Database	It consists of over 5 million characters from 2,200 writers.	<a href="http://unipen.nici.kun.nl">http://unipen.nici.kun.nl</a>
CEDAR Handwriting Database	It consists of lines of text by around 200 writers. Total number of words in the database is 105,573. The database contains both cursive and printed writing, as well as some writing which is a mixture of cursive and printed.	<a href="http://www.cedar.buffalo.edu/handwriting">http://www.cedar.buffalo.edu/handwriting</a>

## Online Handwriting Recognition

Online handwriting is recognizing the handwriting obtained from a digitized pad which provides the trajectory of the pen even when the pen is not in contact with the pad. There are only a few databases only in the public domain.

## Gait Recognition

Gait recognition is identifying the person from the way of walking. The walking style is analyzed from captured

Table 6.

Gait Database	Specifications	WWW Address
NLPR Gait Database	It consists of image sequences captured in an outdoor environment with natural light by a surveillance camera (Panasonic NV-DX100EN digital video camera). The database contains 20 different subjects and three views, namely, lateral view, frontal view, and oblique view with respect to the image plane. The captured 24-bit full color images of 240 sequences have a resolution of 352 by 240.	<a href="http://www.sinobiometrics.com/gait.htm">http://www.sinobiometrics.com/gait.htm</a>
USF Gait Baseline Database	It contains a data set over 4 days, the persons walking in an elliptical path at an axis of (minor $\approx$ 5m and major $\approx$ 15m). The variations considered are different viewpoints, shoe types, carrying conditions, surface types, and even some at 2 differing time instants.	<a href="http://figment.csee.usf.edu/GaitBaseline/">http://figment.csee.usf.edu/GaitBaseline/</a>
CMU Mobo Database	It consists of 25 subjects walking with 4 different styles of walking (slow walk, fast walk, inclined walk, and slow walk holding a ball). The sequences are captured from 6 synchronized cameras at 30 frames per second for 11 seconds.	<a href="http://hid.ri.cmu.edu/HidEval/index.html">http://hid.ri.cmu.edu/HidEval/index.html</a>
Southampton Human ID at a Distance Database	Large Database—The variations captured are: indoor track, normal camera, subject moving towards the left; indoor track, normal camera, subject moving towards the right; treadmill, normal camera; indoor track, oblique camera; and outdoor track, normal camera.  Small Database—The variations captured are: normal camera, subject wearing flip-flops and moving towards the left; normal camera, subject wearing flip-flops and moving towards the right; normal camera, subject wearing a trench coat; and normal camera, subject carrying a barrel bag on shoulder.	<a href="http://www.gait.ecs.soton.ac.uk/database/">http://www.gait.ecs.soton.ac.uk/database/</a>

walking sequences. Details of preparation of gait databases can be found in Nixon, Carter, Cunado, Huang, and Stevenage (1999). The various gait databases available in the public domain are listed in Table 6.

## CONCLUSION

Biometric databases should thus be prepared with a proper sample for enrollment and for verification as well so that the system accuracy can be increased with the decrease in cost of the system. For the other biometric traits, public databases still need to be created on some well-defined user-accepted policies so that there should be a fair evaluation of the algorithms developed by various researchers. The traits which need to be explored for



database creation are ear, hand geometry, palm print, keystroke, DNA, thermal imagery, etc.

## REFERENCES

Burge, M., & Burger, W. (2000). Ear biometrics for machine vision. In *Proceedings of the International Conference on Pattern Recognition* (pp. 826-830).

Campbell, J. P., & Reynolds, D. A. (1999). Corpora for the evaluation of speaker recognition systems. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* (Vol. 2, pp. 829-832).

Daugman, J. (1998). Recognizing persons by their iris patterns. In A. Jain, R. Bolle, & S. Pankati (Eds.), *Biometric: Personal identification in networked society* (pp. 103-121). Amsterdam: Kluwer.

Federal Bureau of Investigation. (1984). *The science of fingerprints: Classification and uses*. Washington, D.C.: U.S. Government Printing Office.

Hong, L., Wan, Y., & Jain, A. K. (1998). Fingerprint image enhancement: Algorithms and performance evaluation. In *Proceedings of the IEEE Computer Society Workshop on Empirical Evaluation Techniques in Computer Vision* (pp. 117-134).

Maltoni, D., Maio, D., Jain, A. K., & Prabhakar, S. (2003). *Handbook of fingerprint recognition*. Springer.

Moghaddam, B., & Pentland, A. (1999). Bayesian image retrieval in biometric databases. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems* (Vol. 2, p. 610).

Monrose, F., & Rubin, A. D. (2000). Keystroke dynamics as a biometric for authentication. *Future Generation Computer Systems*.

Nixon, M. S., Carter, J. N., Cunado, D., Huang, P. S., & Stevenage, S. V. (1999). Automatic gait recognition. In *Biometrics: Personal identification in networked society* (pp. 231-249). Kluwer.

Ratha, N., Chen, S., Karu, K., & Jain, A. K. (1996). A real-time matching system for large fingerprint databases. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8), 799-813.

Vatsa, M., Singh, R., & Gupta, P. (2003). Image database for biometrics personal authentication system. In *Proceedings of CVPRIP*.

Wildes, P. R. (1999). Iris recognition: An emerging biometric technology. *Proceedings of the IEEE*, 85(9), 1348-1363.

Zhao, W., Chellappa, R., Rosenfeld, A., & Philips, P. J. (2000). *Face recognition: A literature survey* (UMD Tech. Rep.).

## KEY TERMS

**Biometric:** A physiological or behavioral characteristic used to recognize the claimed identity of any user. The technique used for measuring the characteristic and comparing it is known as biometrics.

**Biometric Sample:** The unprocessed image or physical or behavioral characteristic captured to generate the biometric template.

**Biometric Template:** This is the mathematical representation of the biometric sample which is finally used for matching. The size of a template varies from 9 bytes for hand geometry to 256 bytes for iris recognition to thousands of bytes for face.

**Enrollment:** The initial process of collecting biometric data from a user and then storing it in a template for later comparison. There are two types: positive enrollment and negative enrollment.

**Failure to Enroll Rate:** Failure to enroll rate is the condition which arises if the biometric sample captured is not of proper quality that a template can be generated from it.

**False Acceptance Rate:** The probability of incorrectly identifying any impostor against a valid user's biometric template.

**False Rejection Rate:** The probability of incorrectly rejecting the valid users or failing to verify the legitimate claimed identity of any user.

**Verification:** Comparing the two biometric templates (enrollment template and verification template) and giving the validity of the claimed identity as to whether both are from the same person or not. This is a one-to-one comparison.

# Business Rules in Databases

**Antonio Badia**

*University of Louisville, USA*

## INTRODUCTION

Though informal, the concept of business rule is very important to the modeling and definition of information systems. Business rules are used to express many different aspects of the representation, manipulation and processing of data (Paton, 1999). However, perhaps due to its informal nature, business rules have been the subject of a limited body of research in academia. There is little agreement on the exact definition of business rule, on how to capture business rules in requirements specification (the most common conceptual models, entity-relationship and UML, have no proviso for capturing business rules), and, if captured at all, on how to express rules in database systems. Usually, business rules are implemented as triggers in relational databases. However, the concept of business rule is more versatile and may require the use of other tools.

In this article, I give an intuitive definition of business rule and discuss several classifications proposed in the literature. I then argue that there is no adequate framework to capture and implement business rules and show some of the issues involved, the options available, and propose a methodology to help clarify the process. This article assumes basic knowledge of the relational data model (Date, 2000).

## BACKGROUND

The term *business rule* is an informal, intuitive one. It is used with different (but overlapping) meanings in different contexts. GUIDE (2000) defined business rule as follows: “A business rule is a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business.” Thus, the rule expresses a policy (explicitly stated or implicitly followed) about some aspect of how the organization carries out its tasks. This definition is very open and includes things outside the realm of information systems (e.g., a policy on smoking). I will not deal with this kind of rule here<sup>1</sup>; instead, I will focus only on business rules that deal with *information*. Intuitively, a business rule in this sense usually specifies issues concerning some particular data item (e.g., what fact it is supposed

to express, the way it is handled, its relationship to other data items). In any case, the business rule represents information about the real world, and that information makes sense from a business point of view. That is, people are interested in “the set of rules that determine how a business operates” (GUIDE, 2000). Note that while data models give the structure of data, business rules are sometimes used to tell how the data can and should be used. That is, the data model tends to be static, and rules tend to be dynamic.

Business rules are very versatile; they allow expression of many different types of actions to be taken. Because some types of actions are very frequent and make sense in many environments, they are routinely expressed in rules. Among them are:

- enforcement of policies,
- checks of constraints (on data values, on effect of actions), and
- data manipulation.

Note: When used to manipulate data, rules can be used for several purposes: (a) standardizing (transforming the implementation of a domain to a standard, predefined layout); (b) specifying how to identify or delete duplicates (especially for complex domains); (c) specifying how to make complex objects behave when parts are created–deleted–updated; (d) specifying the lifespan of data (rules can help discard old data automatically when certain conditions are met); and (e) consolidating data (putting together data from different sources to create a unified, global, and consistent view of the data).

- time-dependent rules (i.e., rules that capture the lifespan, rate of change, freshness requirements and expiration of data).

Clearly, business rules are a powerful concept that should be used when designing an information system.

## Types of Business Rules

Von Halle and Fleming (1989) analyzed business rules in the framework of relational database design. They distinguished three types of rules:

1. **Domain rules:** Rules that constrain the values that an attribute may take. Note that domain rules, although seemingly trivial, are extremely important. In particular, domain rules allow
  - verification that values for an attribute make “business sense”;
  - decision about whether two occurrences of the same value in different attributes denote the same real-world value;
  - decision about whether comparison of two values makes sense for joins and other operations. For instance, EMPLOYEE.AGE and PAYROLL.CLERK-NUMBER may both be integer but they are unrelated. However, EMPLOYEE.HIRE-DATE and PAYROLL.DATE-PAID are both dates and can be compared to make sure employees do not receive checks prior to their hire date.
2. **Integrity rules:** Rules that bind together the existence or nonexistence of values in attributes. This comes down to the normal referential integrity of the relational model and specification of primary keys and foreign keys.
3. **Triggering operations:** Rules that govern the effects of insert, delete, update operations in different attributes or different entities.
  - a. **Definition of business terms:** The terms appearing on a rule are common terms and business terms. The most important difference between the two is that a business term can be defined using facts and other terms, and the common term can be considered understood. Hence, a business rule captures the business term’s specific meaning of a context.
  - b. **Facts relating terms to each other:** Facts are relationships among terms; some of the relationships among terms expressed are (a) being an attribute of, (b) being a subterm of, (c) being an aggregation of, (d) playing the role of, or (e) a having a semantic relationship. Terms may appear in more than one fact; they play a *role* in each fact.
2. **Constraints (also called action assertions):** Constraints are used to express dynamic aspects. Because they impose constraints, they usually involve directives such as “must (not)” or “should (not).” An action assertion has an *anchor object* (which may be any business rule), of which it is a property. They may express conditions, integrity constraints, or authorizations.
3. **Derivations:** Derivations express procedures to compute or infer new facts from known facts. They can express either a mathematical calculation or a logical derivation.

This division is tightly tied to the relational data model, which limits its utility in a general setting. Another, more general way to classify business rules is the following, from GUIDE (2000):

1. **Structural assertions:** Something relevant to the business exists or is in relation to something else. Usually expressed by *terms* in a language, they refer to facts (things that are known, or how known things relate to each other). A term is a phrase that references a specific business concept. Business terms are independent of representation; however, the same term may have different meanings in different contexts (e.g., different industries, organizations, line of business). Facts, on the other hand, state relationships between concepts or attributes (properties) of a concept (attribute facts), specify that an object falls within the scope of another concept (generalization facts), or describe an interaction between concepts (participation facts). Facts should be built from the available terms. There are two main types of structural assertions:

There are other classifications of business rules (Ross, 1994), but the ones shown give a good idea of the general flavor of such classifications.

## **Business Rules and Data Models**

There are two main problems in capturing and expressing business rules in database systems. First, most conceptual models do not have means to capture many of them. Second, even if they are somehow captured and given to a database designer, most data models are simply not able to handle business rules. Both problems are discussed in the following sections. As a background, I discuss how data models could represent the information in business rules. Focusing on the relational model, it is clear that domain constraints and referential integrity constraints can be expressed in the model. However, other rules (expressing actions, etc.) are highly problematic. The tools that the relational model offers for the task are checks, assertions, and triggers.

*Checks* are constraints over a single attribute on a table. Checks are written by giving a condition, which is similar to a condition in a WHERE clause of an SQL query. As such, the mechanism is very powerful, as the condition can contain subqueries and therefore mention other attributes in the same or different tables. However, not all systems fully implement the CHECK mechanism; many limit the condition to refer only to the table in which the attribute resides. Checks are normally used in the context of domains (Melton and Simon, 2002); in this context, a check is used to constrain the values of an attribute. For instance,

```
CREATE DOMAIN age int CHECK (Value >= 18
and Value <= 65)
```

creates a domain named age, expressed as an integer; only values between 18 and 65 are allowed. Checks can be used inside a CREATE TABLE statement, too. Checks are tested by the system when a tuple is inserted into a table (if the check was given within a CREATE TABLE statement, or if the check was given within a CREATE DOMAIN and the table schema uses the domain), or when a tuple is updated in the table. If the condition in the check is not satisfied, the insertion or update is rejected. However, it is very important to realize that when check conditions involve attributes in other tables, changes to those attributes do not make the system test the check; therefore, such checks can be violated. For instance, the following declaration

```
CREATE TABLE dept (
  Dno int,
  Mgrname VARCHAR(50),
  Check (Mgrname NOT IN (Select name from em-
  ployee where salary < 50,000))
```

would block insertions into table dept (or updates of existing tuples) if someone introduced in the tuple a manager that made less than 50,000. However, if someone updated the table emp to change a manager's salary to less than 50,000, the update would not be rejected.

*Assertions* are constraints associated with a table as a whole or with two or more tables. Like checks, assertions use conditions similar to conditions in WHERE clauses; but the condition may now refer to several existing tables in the database. Note that, unlike checks, which are associated with domains or attributes in tables, assertions exist at the database level and are independent of any table. Assertions are checked by the system whenever there is an insertion or update in any of the tables mentioned in the condition. Any transaction must keep the assertion true; otherwise, it is rejected. This makes assertions quite powerful; unfortunately, assertions are

not fully or partially supported by many commercial systems.

*Triggers* are expressions of the form E-C-A, where E is called an event, C is called a condition, and A is called an action. An event is an occurrence of a situation. An event is instantaneous and atomic (i.e., it cannot be further subdivided, and it happens at once or does not happen at all). The typical events considered by active rules are primitive for database state changes (e.g., an insert, delete, or update). A condition is a database predicate; the result of such a condition is a Boolean (TRUE/FALSE). Action is a data manipulation program, including insertions, deletions, updates, transactional commands, or external procedures. In order to allow the action to process data according to the event, most systems use the variables OLD and NEW to refer to data in the database before and after the event takes place. If a row is updated, both OLD and NEW variables have values. If a row is inserted, then only the NEW variable has a value; if a row is deleted, then only the OLD variable has a value.

As a final note, it is necessary to point out that many database developers avoid the use of triggers, if at all possible, for two main reasons: (a) most trigger systems add considerable overhead, and (b) triggers bring in issues of complexity usually associated with programming (i.e., it is hard to ensure that the right semantics are being implemented). However, triggers are very powerful tools and sometimes have no substitute to express business logic (Widom & Ceri, 1996).

## CAPTURING BUSINESS RULES

### Eliciting and Representing Business Rules in Conceptual Models

Because business rules sometimes represent an *implicit* type of knowledge (Krogh, 2000), elicitation of business rules share problems and challenges with knowledge elicitation in general. In particular, complex managerial and organizational issues must be addressed; at the minimum, a managerial action must be taken that encourages the sharing of knowledge by human agents. I will not address elicitation issues here, as they are outside the scope of this article.

Clearly, business rules should be captured in the requirement specification phase of a software project. It is during this phase that real-world information is analyzed and modeled and the scope of the system decided (Davis, 1993). However, there is no single technique that will guarantee capturing all business rules that an organization may need. Most conceptual



models are not adequate for the task, although there have been attempts to develop explicit, declarative representations for business rules. Among them, ORM (Halpin, 1995) provides several mechanisms to specify constraints related to the participation of entities in relationships, which affects some structural and action assertions. Also, an extension for UML (Rumbaugh, Jacobson, & Booch, 1999) has been proposed that addresses expression of constraints: OCL (Object Constraint Language), a semi-formal language that can handle most structural and action assertions (Demuth, Hussmann, & Loecher, 2001). A representation specifically for business rules is Ross's method (Ross, 1994). However, all these representations lack formal semantics, and most of them lack a methodology to indicate how different kinds of rules are to be represented. Finally, none of them addresses the question of how business rules are to be incorporated into the design of an information system.

One of the most well-known and used conceptual models, the entity-relationship (ER) model (Chen, 1976), falls very short of the task of capturing business rules (Badia, 2004). In ER models, there are three basic concepts: entity types, attributes, and relationships. Clearly, the entity types are used to denote business objects, of which rules are predicated. Rules, then, must be expressed in ER models using either attributes or relationships. However, many kinds of rules cannot be expressed with these means: the domain rules of Von Halle and Fleming (1989) actually give us properties of the attributes of the object; because attributes in ER models have no further structure, this information cannot be captured. For example, given an entity type Employee, and an attribute Age, the fact that Age should be between 18 and 65 cannot be expressed in an ER model. Such information can be captured in a data dictionary; however, more complex rules that relate several attributes cannot be added to a dictionary. Assume, for instance, that Employee has two attributes, Salary and Rank, and that each rank determines a maximum and minimum salary for employees in that rank. Such complex relation is unlikely to be written in a data dictionary (under which term, Salary or Rank, should it be included?). As for triggering operations—and the action assertions and derivations of GUIDE (2000)—because they express dynamic aspects and ER models focus on structure but have no dynamic component, such types of rules cannot be represented, either. As an example, assume that a rule exists stating that the salary of an employee can never surpass that of his or her manager. When salaries change, the rule should be used to check the correctness of any increase. However, such rules are not expressible in an ER model. Depending on the type of rule, semiformal techniques such as decision trees, tables, state transition diagrams, or methods in object-

oriented models, can be used. However, business rules are intrinsically connected to the data; therefore, it would be desirable to express them in a formalism that has a direct connection to the conceptual model. Finally, note that information *about* the business rule cannot be captured in any of these formalisms, even though metadata about the business rule (e.g., author, date, sponsor, source, purpose, relation to other rules) is necessary to manage a collection of business rules effectively.

## Expressing Business Rules in Data Models

Even if business rules are somehow captured at requirement specification time, the current process for implementing business rules (usually, to give some specification to a programmer so that an application can be written enforcing the rules) has several drawbacks: The process of creating and codifying a rule is long, labor-intensive, and involves several groups of people, creating a risk for miscommunication; assessing impact of rules (the quality and adequacy of the data produced by its application) is extremely difficult; and changing the rules is extremely difficult.

To decide how to express a given business rule, the most promising approach is to classify the rule according to some characteristic that will allow one to determine an efficient implementation. There are some obvious options, such as classifying rules as static or dynamic or classifying rules according to the data they affect.

The first classification makes sense because some rules are more intimately bound to change than others. For instance, a rule stating that salary increases for employees cannot exceed 10% is clearly dependent on a change. Therefore, I define dynamic rules as those that are *directly* related to change, and static rules as those that are not. Even though this definition is somewhat ambiguous, it can be made more precise with a test. To specify the rule, is there a need to refer to the before-and-after of a certain action? If one applies the test to the examples, the rule about employees' salaries not being higher than the managers can be seen as static, while the rules about employees salaries not increasing by more than 10% can be seen as dynamic.

As for classifying rules according to the data they affect, to make such a distinction meaningful in the context of a database the classification must be based on the conceptual model used to design the database, one can then talk about rules affecting a single attribute, several attributes within a single entity, several attributes in several entities, and relationships, which implicitly means the related entities. There is good reason

to look at the scope of the rule: The scope of the tools at our disposal (i.e., checks, assertions, and triggers) is different.

Finally, another important difference among the rules is whether they state a “must” situation or a “should” situation. A must rule is one that the situation cannot be changed to accommodate a violation, while a should rule is one where things can be changed to deal with rule violations. For instance, if the rule about an employee’s salary not being higher than a manager’s is considered a must rule, any attempts to raise an employee’s salary or lower a manager’s salary have to be outright rejected, and if the rule is considered a should rule, changes can be made. To raise an employee’s salary above a manager’s, the manager’s salary can be raised, too, to preserve the rule. The difference is significant because the system needs to know what options, if any, to offer when rules are violated.

Because our goal is to decide how to implement the rules, we must also take into account the characteristics of the tools available. Table 1 summarizes some important differences among triggers, assertions, and checks.

While triggers and assertions can mention any table and any attribute in the database, checks are limited to the domain or table in which they are declared. Also, checks are not always guaranteed to hold (see example in the previous section), whereas the system guarantees that assertions and triggers will always be enforced. Finally, when a check or an assertion are violated, the transaction that violates them is rejected; when a trigger is fired because a condition it checks for holds (usually a violation of a rule), the action part of the trigger allows a programmer to specify exactly how to react.

The previous observations about rules and tools lead me to propose the following classification in order to guide implementation. The classification does not claim to be optimal; it simply responds to the characteristics of the tools at one’s disposal.

- **Static (must) rules:** These rules involve a state change, *and* any transaction that invalidates them must be outright rejected. I further subdivide these rules as follows:
  - **Rules that affect a single attribute:** These may include some domain rules noted in von Halle and Fleming (1989) and the definitions of business terms noted in GUIDE (2000). These rules should

be implemented with a check. As an example, consider the previous rule about employees’ ages. Implementing such rules as checks is advantageous because it is simple (i.e., checks are easy to write), natural (i.e., the check can be attached to the domain of the attribute in question or to its table), and, in the case where no other attributes are present, guaranteed to hold.

- **Rules that affect several attributes in one entity:** These may include some domain rules and some integrity rules noted in Halle (1998) and some facts and action assertions noted in GUIDE (2000). These rules should be implemented with an assertion, because even though checks can mention attributes in other tables, they are not guaranteed to hold (see the example in the previous section).
- **Rules that affect several attributes in several entities:** These may include some integrity rules noted in Halle (1998) and some action assertions noted in GUIDE (2000). These rules should be implemented with an assertion, for the same reasons as in the previous case. Assume, for instance, that there are ranks in the organization and each employee has a rank. Associated with each rank there is some information; for instance, the maximum and minimum salary for that rank. An obvious rule is that employees’ salaries remain within what is dictated by each employee’s rank. Then the assertion would have to mention both tables:

```
CREATE ASSERTION salary-rank
(NOT EXISTS (SELECT * FROM EMPLOYEE, RANK
            WHERE Employee.rankid
            =RANK.rankid AND
            (Employee.salary < rank.min_salary OR
            Employee.salary > rank.max_salary))).
```

- **Dynamic (should) rules:** These are rules that involve state changes or that may respond to a violation in several ways, including the triggering operations noted in Halle (1998) and the derivations noted in GUIDE (2000). I can also further subdivide them as was done for static rules; however, in a relational database all such rules must be written as triggers. This is because dynamic rules may have to mention the before-and-after of an action, possible

Table 1. Differences among relational database tools

Difference	Trigger	Assertion	Check
Scope	Unlimited	Unlimited	Limited
Guaranteed to always hold?	Yes	Yes	No
Allows specifying options	Yes	No (always rejection)	No (always rejection)



in a trigger but not possible in a check or an assertion, and “should” rules need to specify how to react to a violation, for which checks and assertions have no proviso, but triggers have the action part. For instance, the rule stating that employees’ salaries must remain within what is dictated by the employees rank can be written as<sup>2</sup>:

```
EVENT: UPDATE(EMPLOYEE)
CONDITION: (NOT EXISTS (SELECT *
                    FROM RANK WHERE rankid =
NEW.rankid AND
    (NEW.salary > TEMP.max_salary OR NEW.salary
< min_salary)
ACTION: ...
```

If this is a should rule, several things can be done when the rule is violated; rejecting the salary change is only one of them. The desired behavior is programmed into the action part. Note that there is also more than one way in which the rule can be violated: not only by changing an employee’s salary, but also by changing an employee’s ranking or changing a ranking’s maximum or minimum salary. Hence, multiple triggers should be written, unless the system allows compound events to be expressed.

As an example of a typical dynamic rule, use the old example of salaries not increasing by more than 10%. In this case, the trigger would look like this:

```
EVENT: UPDATE(EMPLOYEE)
CONDITION: OLD.salary * 1.1 > NEW.salary
ACTION: ...
```

As stated previously, the use of triggers is limited in many cases. Using an approach like the one proposed here, the use of triggers may be avoided in some cases.

Although implementation issues are not mentioned here, they are an important aspect of the problem because, as stated previously, trigger systems add substantial overhead to database processing. It is also likely that assertions are not implemented in commercial systems because of the difficulty of supporting them efficiently. Both assertions and triggers interact with the transaction processing part of the system, raising difficult questions, such as, When should a trigger or assertion be checked? What should be done when more than one assertion or trigger should be executed? Does order matter? This is an area that is still the subject of considerable research (Widom & Ceri, 1996).

## FUTURE TRENDS

To implement business rules in a relational database, it is necessary not only to find some way to capture and express the rule in the requirement specification, perhaps using a combination of techniques (and surely going beyond what usual conceptual models have to offer), but also to find some way of implementing the rule in the database, a process that might get complicated. It is clear that checks, assertions, and triggers together manage to cover most of the usual business rules; however, it is difficult to decide which tool to use for a particular rule. It therefore would be desirable to develop either a new construct specifically tailored to business rules or to extend the existing rules to make the task more manageable. From the analysis used here, some obvious avenues of development can be pointed out. For instance, extending checks and assertions with the ability to specify what to do in case a condition is violated or ensuring that a check is always guaranteed to hold would allow many static rules to be expressed without having to resort to triggers, which could then be used only if truly necessary.

## CONCLUSION

Although business rules are a very important part of information systems, there is little academic research on them. Some practitioners have contributed to the understanding of the subject, but there is no underlying framework for its study. Also, support for business rules in conceptual models and database systems leaves much to be desired. As a result, eliciting, expressing, and capturing business rules remain more of an art than a science. However, given their practical importance, more research in the subject is expected in the near future.

## REFERENCES

- Badia, A. (2004, March). Entity-relationship modeling revisited. *ACM SIGMOD Record*, 33(1), 77-82.
- Chen, P. (1976). The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems*, 1(1), 9-36.
- Date, C. (2000). *An introduction to database systems* (7th Ed.). Reading, MA: Addison-Wesley.
- Davis, F. (1993). *Requirement specification: Objects, functions and states*. Upper Saddle River, NJ: Prentice Hall.

Demuth, D., Hussmann, H., & Loecher, S. (2001). OCL as a specification language for business rules in database applications. In M. Gogolla & C. Kobryn (Eds.), *Proceedings of the 4<sup>th</sup> International UML Conference: Vol. 2185* (pp. 104-114). New York: Springer.

GUIDE (2000). *The GUIDE business rules project: Final report*. Business Rules Group.

Halle, B. von, & Fleming, C. (1989). *Handbook of relational database design*. Reading, MA: Addison-Wesley.

Halpin, T. (1995). *Conceptual schema and relational database design I* (2nd Ed.). Upper Saddle River, NJ: Prentice Hall.

Krogh, G. (2000). *Enabling knowledge creation*. UK: Oxford University Press.

Melton, J., & Simon, A. R. (2002). *SQL 1999: Understanding relational language components*. San Francisco: Morgan Kaufmann.

Paton, N.W. (Ed.). (1999). *Active rules in database systems*. New York: Springer.

Ross, R. (1994). *The business rule book: Classifying, defining and modeling rules*. Database Research Group.

Rumbaugh, J., Jacobson, I., & Booch, G. (1999). *The unified modeling language reference manual*. Reading, MA: Addison-Wesley.

Widom, J., & Ceri, S. (Eds.). (1996). *Active database systems: Triggers and rules for advanced database processing*. San Francisco: Morgan Kaufmann.

## KEY TERMS

**Action:** Part of a trigger that is executed when the condition in the trigger is evaluated and found true. This is the most open-ended part of a trigger, because different database vendors have allowed different functionality in their implementations. At a minimum, data manipulations operations (e.g., insert, delete, update) are allowed.

**Assertion:** Expression in a relational database system that allows stating a condition involving several attributes and several tables. Assertions are first-class database entities, such as tables.

**Business Rule:** Statement that defines or constrains business objects, their behavior, and relationships. Usually expressed in a semiformal language, using a vocabulary of business terms and verbs such as *have to*, *should*, and *must*.

**Business Term:** Word or expression denoting a concept that has a particular meaning in the context of an enterprise.

**Check:** Expression in a relational database system that allows stating a condition involving an attribute in a table; used with CREATE DOMAIN or CREATE TABLE statements.

**Condition:** Part of a trigger that is evaluated when the event in the trigger takes place. It is usually a predicate in SQL that evaluates to true or false.

**Event:** Part of a trigger that refers to the occurrence of a certain situation. When the event occurs, the rule is triggered (i.e., is scheduled so that its condition is evaluated). In most systems, the event can mention only basic database actions (e.g., insertions, deletions, updates).

**Trigger:** Rule of the form Event-Condition-Action, used in relational database systems to implement active functionality.

## ENDNOTES

<sup>1</sup> Also, managerial issues (enforcing the rules, setting business policies, etc.) are not dealt with here.

<sup>2</sup> Because different systems implement triggers in very different ways, I use a stylized syntax that makes clear the EVENT, CONDITION, and ACTION of each trigger. Because expressiveness is the main interest, implementation issues that surround the use of triggers are not dealt with. In particular, and to simplify, it is assumed that every trigger is applied to the database immediately after the event takes place. Within the body of the trigger, the words OLD and NEW refer to the affected tuple(s) values before and after the event is applied to the database.

# Business-to-Business Integration

**Christoph Bussler**

*Digital Enterprise Research Institute (DERI), Ireland*

## **BUSINESS-TO-BUSINESS (B2B) INTEGRATION TECHNOLOGY**

Businesses worldwide started exchanging electronic business messages with each other around 1970. This coincides with the establishment of wide-area computer networks. Businesses realized the potential immediately to send electronic messages instead of paper letters in order to conduct business electronically. Computer networks provided a significant increase of transmission speed, less failures due to message losses, and direct processing of messages upon receipt without manual transcript from paper to computer terminals or vice versa. Overall, business interactions became a lot more reliable and efficient.

The direct processing capability of electronic messages enabled the seamless integration of the messaging environment and the back-end application system processing. The electronic integration of businesses was achieved this way and the technological basis was called business-to-business (B2B) integration technology. However, B2B integration technology was not affordable to every business independent of its size due to its high deployment and maintenance cost. Therefore, only big businesses could afford electronic B2B integration. Smaller businesses continued to rely on paper letters or fax transmissions.

Today it is commonly accepted to exchange electronic messages between businesses. Current trends are to make B2B technology more accessible to every business independent of its size through ubiquitous Web technology. This new technology is called Web services and all software vendors readily provide solutions (Alonso, Casati, Kuno, & Machiraju, 2004; Bussler, 2003).

## **BACKGROUND**

Dedicated networks called value added networks (VANs) started providing the transport of electronic messages between businesses around 1970. Since their service is very reliable they still are extremely popular today and widely used. VANs do not support direct communication between businesses. Instead, businesses have to upload and download messages asynchronously from VANs. VANs provide a persistent mailbox system where every partner has a dedicated mailbox storing the electronic

messages addressed to it. Based on their behavior VANs are an asynchronous network for exchange of messages. In addition (“value added”), they provide services like storage, backup, and billing.

Through the emergence of VANs, business-to-business (B2B) integration was born (Bussler, 2003). VANs made a big difference in competitiveness since businesses could rely on the extreme high speed of electronic data transmission compared to paper-based communication through postal services. Different industries embarked on B2B integration through, at that time, a new form of communication.

It became immediately apparent that it is not advantageous at all for businesses to define their own message formats and send them over VANs. This would require a business to deal with all the different message formats defined by all its business partners if each one would provide its own message format. A significantly better approach was to standardize the message formats across businesses so that a business could use the same message formats across all its business partners from the same industry. In specific industries, message definition standards have been developed for over 30 years now. An example for the supply-chain industry is EDI (ASC, 2004), for the banking industry is SWIFT (2004), and for the insurance industry is ACORD (2004). These standards are called B2B standards and enable the fast integration of a business with others in the same industry. A business can comply with the standard and by that is ensured that the message exchange with the other partners is interoperable. The standardized message formats ensure interoperability.

While the message formats have been standardized, the technical software implementations for B2B communication have not. Every VAN is providing its own communication software for uploading and downloading the messages from the mailboxes. In order to allow messages to be automatically processed by back-end application systems, businesses had to integrate their VANs’ communication software into their information system environment. This back-end application system connectivity is necessary for retrieving and storing the data from the back-end application systems that are communicated with through the communication software. That meant that each business had to do custom implementation in order to make the B2B communication with its partners work.

## **INTEGRATION TECHNOLOGY**

In the late 1980s and early 1990s, the first B2B integration products appeared on the software market. These products were off-the-shelf software that did not require any custom coding by the businesses anymore in order to either connect to the VAN's software or the back-end application system software. Instead, it could automatically link the VANs' communication software and the back-end application systems (like, for example, enterprise resource planning, ERP, systems) within the businesses. Establishing B2B integration became more and more like a turn-key software solution instead of a custom coding task.

This automatic connectivity to back-end application systems is achieved through specialized software adapters that adapt the B2B integration product's interface to the particular interface of the back-end application system (Apte, 2002). An adapter enables the B2B integration technology to insert as well as to extract data from the back-end application system. The benefit for the businesses was that they did not have to perform any custom coding anymore but could use prepackaged B2B integration software instead. That allowed buying the integration functionality without going through internal software development processes. A standard called J2EE Connector Architecture (JCA, 2004) allows software vendors to build standardized adapters. This contributes to the turn-key nature of integration products.

The main problem of B2B integration, besides integrating back-end application systems and the communication software to connect to VANs or the Internet, is data definition mismatches. An example of a data definition mismatch is that an address can be defined as a single string or as a structured record, where each address element like city or zip code is stored in a separate field. Mediation (or, synonymously termed, transformation) overcomes the data definition mismatch problem that exists between B2B standards and the data models of back-end application systems. The same piece of data can be represented very differently in a B2B standard and in a back-end application system. When data is extracted from the back-end application system in its format, it has to be reformatted to the message format of the B2B standard. This reformatting is called mediation (Bussler, 2003; Omelayenko & Fensel, 2001; Rahm & Bernstein, 2001). Not only the structure of the data might have to be changed (e.g., from a one-string representation into a structured record), but also the content representation (e.g., "Ireland" in one representation has to be replaced by "IRL" in another one). The change has to happen in such a way that the meaning of the data (data semantics) is not changed at all, i.e., the data semantics have to be absolutely preserved.

Not only data structures and vocabularies can mismatch, but also the message exchange sequences. For example, while a back-end application system might send one message and expects a return message back, a B2B interaction might have more messages, including acknowledgement messages, in order to establish an exactly once transmission. These differences in message exchanges are overcome by process mediation (Fensel & Bussler, 2002). Process mediation ensures that all communicating parties receive the messages in the sequence they require.

When businesses send and receive messages they have to make sure that they know their communication partners. Otherwise they might accept messages that are not sent by a contractually bound partner. If this is the case then messages might be processed that should not. On the other hand, messages must only be sent to partners, not to any other organization. In order to define partners as well as their specific properties, standards like collaboration protocol profile/ agreement (CPP/A) are established (ebXML, 2004).

Adaptation, data and process mediation, as well as partner management are the most essential concepts that integration technology has to implement.

## **CURRENT DEVELOPMENTS**

The availability of software products for B2B integration lowered their price significantly, mainly because custom coding was not necessary any more. This enabled smaller business to participate in B2B integration and soon more businesses made use of the B2B integration technology then ever before.

In the mid-1990s XML (XML, 2004) emerged as viable technology and was picked up by software developers around the world. At the same time, the first B2B integration products based on XML as the message syntax were formed and appeared on the market. The promise was that the new technology was going to make the B2B integration task a lot easier and a lot cheaper. However, this was not the case as the same integration problems like adaptation, mediation, or partner management had to be solved, just on a different technological basis. Mediation, adapters, connectivity to network software, etc. all require a solution in the context of XML-based integration technology, too.

The Internet was available as a communication platform and many B2B integration products started using the Internet instead of VANs. That proved difficult since the open and unreliable Internet required the development of secure and reliable communication protocols. This is in contrast to VANs that are proprietary networks and their access is restricted to the VANs' customers. In addition,



there was not asynchronous behavior of the network anymore. Instead, businesses had to communicate directly with each other. This required sufficient uptime to avoid communication errors due to unavailability.

Recently new B2B integration standards like RosettaNet (2004), cXML (2004), and ebXML (2004) have appeared. These use XML as the syntax for defining message types. However, XML did not deliver on the promise that standards became better or easier to use. The promise was delivered on the assumption that a commonly accepted message syntax will make a significant difference. Instead, the contrary happened. Because it was so easy to define message types in XML, too many were created, quickly contradicting the purpose of standards as such. Instead of less variety of message types, a huge number appeared. The flurry of proposed XML message standards made potential customers wonder about how these are going to be managed in the long run. Consequently, customers held back in investing into XML-based B2B technology and continued to use VANs instead.

XML and the formats used earlier for B2B standards do not capture the semantics of the data, just the structure (syntax) of the data. Because of this situation it is not possible for the B2B integration technology to semiautomatically or fully automatically perform the mediation. Instead, a developer has to develop the rules that define the mediation by hand. While this is a possible solution, manual encoding of the rules often leads to severe mistakes in the B2B communication.

Independent of its usefulness, once XML was “discovered” it became clear that XML messages can be sent through HTTP connections. The sending of XML messages through Internet technology is called Web services. Prominent standards in the space are SOAP (2004) and WSDL (2004). Both together allow defining basic and single message exchanges between a client and a server. Soon it was discovered that single message exchanges are insufficient for serious communication patterns between businesses. Several so-called process standards try to address this like BPEL (2004), ebXML BP (2004), and WSCDL (2004) as well as many research groups.

## **FUTURE TRENDS**

The advent of Semantic Web technology promises to improve the overall situation significantly, especially providing solutions for the mediation problem. Ontology languages like RDF (2004), RDFS (2004), and OWL (2004) allow capturing semantics about data. Message formats can be defined with ontology languages not only describing the syntax but also the semantics. Once the semantics are formally captured, the mediation problem can be addressed a lot better. This is possible since the semantics

are formally denoted and the integration technology can infer from this if two data definitions match or not. If they do, transformation between the two can be automatically achieved. If not, a human can be brought into the loop, helping to overcome the semantic mismatch. However, solving the transformation problem based on ontologies is a very recent development and no products can be bought yet that have this higher-level functionality.

A significant effort applying Semantic Web technology in order to solve the integration problems is the Web Service Modeling Ontology (WSMO, 2004). This ontology, which is developed by a significant number of organizations, uses Semantic Web technology to define a conceptual model to solve the integration problem that encompasses B2B integration. The main conceptual elements proposed are ontologies for defining data and messages, mediators for defining transformation, goals for dynamically binding providers, and Web services for defining the interfaces of communicating services. In order to define a specific B2B integration, the formal Web Service Modeling Language (WSML, 2004) is defined. This can be processed by a WSML parser. In order to execute B2B integration the Web Service Modeling Ontology Execution environment (WSMX, 2004) is developed. It serves as a runtime environment for integrations defined through WSML.

## **CRITICAL ISSUES OF B2B INTEGRATION TECHNOLOGIES**

Despite the widespread use of B2B integration in industry and governmental organizations, B2B integration technology is still an area of major product development efforts. Software vendors like BEA (2004), IBM (2004), Microsoft (2004), and Oracle (2004) continue to develop new products providing B2B integration functionality.

In recent years the scientific community picked up the topic of integration in the context of Web services (2004) and Semantic Web activities (Fensel & Bussler, 2002); see also future trends above.

Independent of the specific approaches in industry and academia, critical issues, as listed in Table 1, have to be addressed for the technology to work and for the research to make significant contributions.

## **CONCLUSION**

Business-to-business (B2B) integration is absolutely essential for current businesses and organizations to not only stay competitive but also gain market share. Furthermore, the trend is going towards connecting all

Table 1. A summary of critical issues

<p><b>Choreography</b> Message exchange pattern that a business exposes and that a business partner has to follow for successful business message exchange.</p> <p><b>Communication software</b> Software that implements the communication of messages between businesses based on a specific network protocol like TCP/IP.</p> <p><b>Conversation</b> Series of message exchanges between two or more businesses in order to achieve a business goal.</p> <p><b>Error management</b> Handling of error messages that indicate an error by executing subsequent actions to neutralize the error situation into a consistent business state.</p> <p><b>Exactly-once message transmission</b> Assurance that each sent message is received either once or not at all by a business partner in presence of a non-reliable and non-transactional network protocol.</p>	<p><b>Mediation</b> Semantics-preserving transformation of a message in a message format into a separate message following a different message format.</p> <p><b>Message security</b> Insurance that message cannot be read by unauthorized businesses and that messages are not replayed, deleted, or modified while in transmission.</p> <p><b>Non-repudiation</b> Security functionality that ensures that the sender and the receiver cannot claim to not have received or sent a specific business message.</p> <p><b>Orchestration</b> Sequenced invocation of several business partners by a given business partner for it to achieve its business goals.</p> <p><b>Partner identification</b> Unique identification of business partners, including the assurance that they do not assume a wrong identity.</p>
---	---

enterprises electronically for the benefit of the enterprises as well as customers. Once all business are connected, all business interactions can be implemented as business messages, making the interactions as efficient as possible.

Ultimately the Semantic Web technology will make the task of integrating businesses easy due to the semantic description of the message formats as well as choreographies and orchestration. Once semantically described, the goal of self-composing enterprises becomes feasible.

**REFERENCES**

ACORD (2004). ACORD Corporation. [www.acord.org](http://www.acord.org)

Alonso, G., Casati, F., Kuno, H., & Machiraju, V. (2004). *Web services: Concepts, architectures and applications*. Berlin, Germany: Springer-Verlag.

Apte, A. (2002). *Java Connector Architecture: Building enterprise adaptors*. Indianaapolis, IN: Sams.

ASC (2004). The Accredited Standards Committee (ASC) X12. [www.x12.org](http://www.x12.org)

BEA (2004). BEA Corporation. [www.bea.com](http://www.bea.com)

BPEL (2004). OASIS Web Services Business Process Execution Language. [www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wsbpel](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel)

Bussler, C. (2003). *B2B integration*. Berlin, Germany: Springer-Verlag.

cXML (2004). Commerce XML. [www.cxml.org/](http://www.cxml.org/)

ebXML (2004). Electronic Business using eXtensible Markup Language (ebXML). [www.ebxml.org](http://www.ebxml.org)

ebXML BP (2004). OASIS ebXML Business Process. [www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=ebxml-bp](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ebxml-bp)

Fensel, D., & Bussler, C. (2002). The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications*, 1(2).

IBM (2004). International Business Machines Corporation. [www.ibm.com](http://www.ibm.com)

JCA (2004). J2EE Connector Architecture. [java.sun.com/j2ee/connector/index.jsp](http://java.sun.com/j2ee/connector/index.jsp)

Microsoft (2004). Microsoft Corporation. [www.microsoft.com](http://www.microsoft.com)

Omelayenko, B., & Fensel, D. (2001). An analysis of integration problems of XML-based catalogues for B2B e-commerce. In *Proceedings of the ninth IFIP 2.6 Working Conference on Database (DS-9): Semantic Issues in E-commerce Systems*. New York: Kluwer.

Oracle (2004). Oracle Corporation. [www.oracle.com](http://www.oracle.com)



OWL (2004). Web Ontology Language. [www.w3.org/2001/sw/WebOnt/](http://www.w3.org/2001/sw/WebOnt/)

Rahm, E., & Bernstein, P. (2001). A survey of approaches to automatic schema matching. *VLDB Journal*, 10.

RDF (2004). Resource Description Framework. [www.w3.org/RDF/](http://www.w3.org/RDF/)

RDFS (2004). Resource Description Framework Schema. [www.w3.org/TR/rdf-schema/](http://www.w3.org/TR/rdf-schema/)

RosettaNet (2004). Rosettanet. [www.rosettanet.org](http://www.rosettanet.org)

SOAP (2004). Simple Object Access Protocol (SOAP) 1.2. [www.w3.org/TR/soap12-part1/](http://www.w3.org/TR/soap12-part1/) and [www.w3.org/TR/soap12-part2/](http://www.w3.org/TR/soap12-part2/)

SWIFT (2004). Society for Worldwide Interbank Financial Telecommunication (SWIFT). [www.swift.com](http://www.swift.com)

Web Services (2004). Web Services Activity. [www.w3.org/2002/ws/](http://www.w3.org/2002/ws/)

WSCDL (2004). Web Services Choreography. [www.w3.org/2002/ws/chor/](http://www.w3.org/2002/ws/chor/)

WSDL (2004). Web Service Description Language (WSDL) 1.1. [www.w3.org/TR/wsdl](http://www.w3.org/TR/wsdl)

WSML (2004). Web Service Modeling Language. [www.wsmo.org/wsml](http://www.wsmo.org/wsml)

WSMO (2004). Web Service Modeling Ontology. [www.wsmo.org](http://www.wsmo.org)

WSMX (2004). Web Service Modeling Execution Environment. [www.wsmx.org](http://www.wsmx.org)

XML (2004). Extensible Markup Language. [www.w3c.org/XML/](http://www.w3c.org/XML/)

## KEY TERMS

**B2B Integration Technology:** A software system that provides business-to-business integration functionality by sending and receiving messages and retrieving and storing them in back-end application systems.

**Back-End Application System:** A software system that manages business domain-specific data for businesses, e.g., enterprise resource planning (ERP) systems.

**Business Partner:** A business partner is a company or organization with which to engage in a business relationship, e.g., buyer, seller, or shipper.

**Choreography:** Choreography is the message exchange behavior that a business exposes in order to participate in a business relationship based on electronic message exchange.

**Mediation:** Mediation is the semantics-preserving transformation of a message in one message format into a message of a different message format that represents the same information.

**Message Standard:** A message standard defines the attributes as well as possible values that a business has to use in order to support interoperable electronic message exchange.

**Orchestration:** A business uses orchestration in order to define the electronic message interaction with other business partners in order to fulfill its obligations.

**Value Added Network:** A value added network is a company that provides network capabilities to business partners that want to exchange electronic messages. A VAN also provides storage, backup, security, and other services in order to provide more message exchange functionality.

# CASE Tools for Database Engineering

**Jean-Luc Hainaut**

*University of Namur, Belgium*

**Jean Henrard**

*REVER S.A., Belgium*

**Jean-Marc Hick**

*REVER S.A., Belgium*

**Didier Roland**

*REVER S.A., Belgium*

**Vincent Englebert**

*University of Namur, Belgium*

## INTRODUCTION

Designing and implementing a database comprising a few tables require a level of expertise that is readily found among most experienced users, provided they are somewhat keen on office productivity tools. Playing a dozen of hours with Microsoft Access should give clever and motivated users sufficient feeling and technical skill to develop small workable databases.

However, things quickly get harder for larger databases, particularly when they have to be integrated in different environments and to meet the needs of several applications. Very large and complex databases often include distributed heterogeneous components that have been developed by dozens of more or less skilled developers at different times, with different technologies and through different methodologies (or absence thereof). Mastering, integrating, maintaining, renovating, evolving, and migrating such complex systems and developing new components for them are highly complex engineering processes that are far beyond the capacity of individuals, whatever their experience and skill. Such problems cannot be addressed without the help of robust methodologies, supported by powerful tools. The following conservative figures illustrate the need for a disciplined, tool-based, approach to database engineering. Considering that (1) each conceptual entity type should ideally be accompanied by a two-page documentation giving its precise semantics and properties, (2) each entity type, together with its attributes and relationship types, translates into two tables on average, and (3) each table is coded by 150 lines of SQL statements, including structural and active components; a standard conceptual schema comprising 1,000 entity types will

be described by a 2,000-page documentation and will produce 300,000 lines of SQL-DDL code. Since database engineering is a special branch of software engineering, the concept of database-centered computer-aided software engineering tools—DB CASE tools for short—appears quite naturally. As a matter of fact, most CASE tools include a strong component intended to help developers design and produce (mainly relational) databases.

This paper discusses some important aspects and functions of DB CASE tools through the description of DB-MAIN, a programmable general-purpose CASE tool that has been designed to support most complex engineering processes for large databases. In particular, it includes, besides standard functions, powerful capabilities for reverse engineering and federated database building.

## BACKGROUND: DATABASE ENGINEERING REQUIREMENTS

Database engineering mainly deals with data structures, data and code related to one or several databases, all of which must be adequately documented in order to be exploited, maintained, and modified in a reliable and efficient way all along the lifetime of these databases. The core of the documentation of a database is a hierarchy of schemas, each of which describes in a precise way its information/data structures at a definite level of abstraction. Most design approaches agree on a four-level architecture comprising a conceptual schema, a logical schema, a physical schema, and code (Batini, Ceri, & Navathe, 1992). The *conceptual schema* describes the information structures that are of interest in

the application domain in a formalism that is technology-independent, such as the entity-relationship model. The *logical schema* is the translation of the conceptual schema according to the model of a family of DBMSs. The *physical schema* adds to the logical structures DBMS-specific technical constructs intended to give the database such properties as efficiency and reliability. The *code* translates the physical schema into a compilable DDL program.

Once a database has been deployed in its production environment, it enters a maintenance and evolution cycle, through which its schema (and consequently its contents) is continuously modified to follow the evolution of functional, organizational, and technological requirements. Both modern and legacy databases are migrated to more advanced platforms; they are integrated to form federations, generally through wrapping and mediation techniques; and they are interfaced with the Web and feed data warehouses. These processes all require the availability of a precise, up-to-date documentation of the concerned databases. Whenever this documentation is missing, the databases must be redocumented, a process called *reverse engineering*.

DB CASE tools are to support these processes, ranging from traditional analysis and design to reverse engineering and federation. They collect, store, and manage not only schemas and their interrelationships (ensuring traceability), but also pieces of code and multimedia documentation related to each step of the database life cycle.

A large organization can include several hundreds of databases, each described by several schemas. Each schema appears to be a graph, made up of hundreds of thousands of vertexes (abstracting entity types, attributes, relationship types and roles, constraints, etc.) This size factor implies specific representation modes and powerful tools to consult, analyze, and transform schemas. Large collections of databases require several design and development teams that call for collaborative work support. Finally, no two organizations are alike and share the same methodological culture, so CASE tool customization and extendability are often highly desirable.

In the next section, we will discuss some important functions of DB CASE tools that derive from these requirements.

## FUNCTIONS OF CASE TOOLS

### Schema Management

This is the basic goal of any CASE tool, namely, allowing developers to enter schemas, including through

graphical interfaces; to store them into a schema database, called a *repository* or encyclopedia; to consult, browse through, and query them; to view them through adequate visualization modes; to output printable documentation; and to exchange specifications with other CASE tools, notably through interchange formats such as XMI (Object Management Group, 2003).

Schemas are expressed into a specific model, depending on the abstraction level and the concerned technology. At the conceptual level, most tools rely on some sort of entity-relationship model. The UML class diagrams belong to this family and are gaining increasing popularity. However, their weaknesses and idiosyncrasies make them a poor support to express large and complex schemas, specially in nonstandard processes such as reverse engineering.

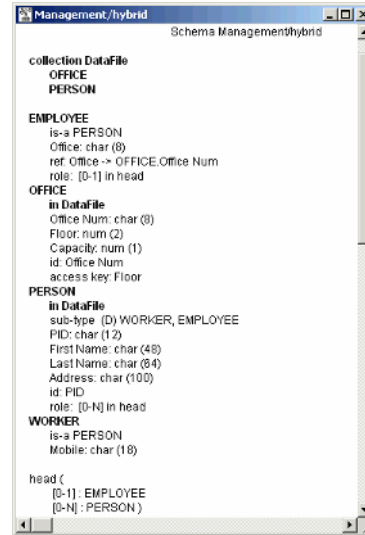
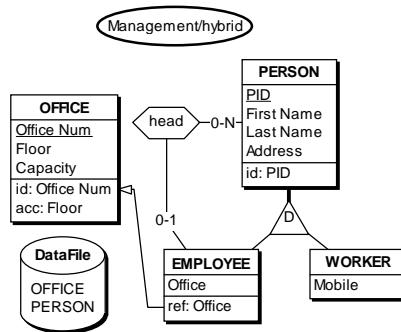
DB-MAIN is based on the wide-spectrum model GER (generic Entity-relationship) that encompasses most commonly used models, whatever their abstraction level and their technology (Hainaut, 1996). An operational model *M* is defined by a specialization mechanism through which the constructs of the GER pertinent in *M* are selected and renamed, and the assembly rules for valid *M* schemas are stated. The GER includes such constructs as entity types, domains, attributes, relationship types, methods, and a large variety of integrity constraints. It also comprises logical and technical constructs, such as foreign keys, access paths, access keys, and files. The tool includes the rules for the most popular models such as Entity-relationship, extended UML class diagrams (DB-UML), relational, object-oriented, CODASYL, COBOL files, and XML (DTD and Schemas). Additional personalized models can be defined thanks to a method engineering environment. Figure 1 shows a hybrid schema that includes constructs from several technologies and several abstraction levels. Such a schema is usual in ongoing reverse engineering projects.

DB-MAIN offers several ways to query and browse through a schema. In addition to six customizable views of schemas, two of them being illustrated in Figure 1, objects can be selected by name patterns, keywords in object annotations, and structural patterns. Selected objects can be marked in order to be retrieved later or to be passed as arguments to further processes.

### Schema Checking and Validation

Depending on the methodological standard adopted or the target technology, schemas are to meet specific requirements in order to be valid. In particular, a schema can be analyzed against a definite model to check whether it complies with it. For instance, a conceptual schema that has been transformed into XML structures should be validated against the XML schema model before passing it to the code generator.

Figure 1. Graphical and hypertext views of a hybrid DB-MAIN schema. They show examples of entity types, IS-A hierarchy, attributes, and relationship types and attributes, but also logical constructs such as a foreign key as well as physical constructs (data file and access key).



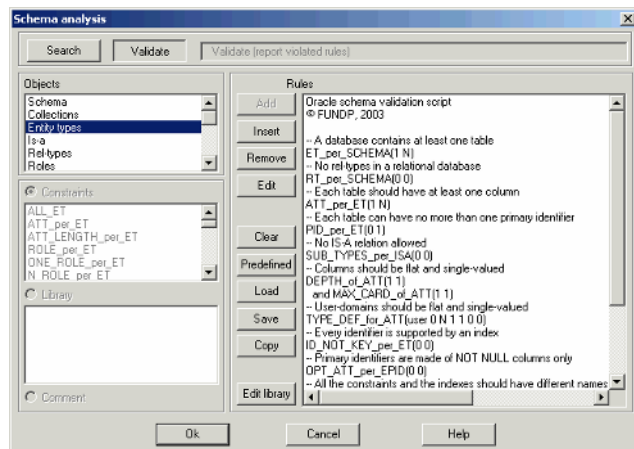
DB-MAIN includes a Schema Analyzer (see Figure 2) that provides three main functions. First, it allows the designer to build, save, and reuse specific models as a set of rules defined by structural predicates. Secondly, its engine can analyze the current schema against a specific model and identify (report, select, or mark) the schema objects that violate any rule of this model. Conversely, it can identify all the objects of the current schema that meet at least one of the rules of a model. DB-MAIN includes 12 predefined models, ranging from COBOL file structures

and XML physical models to ERA and DB-UML conceptual models.

## SCHEMA TRANSFORMATION

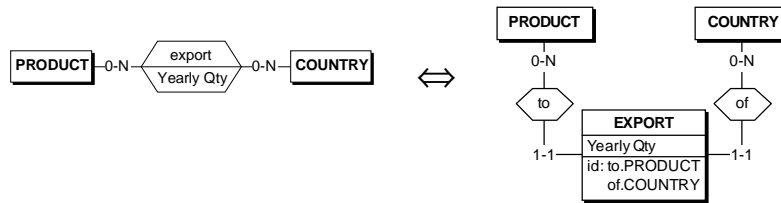
Most database engineering processes intrinsically consist in producing specifications from existing ones, an activity that can be described as schema transformations (Hainaut, 2005; Rosenthal & Reiner, 1994; van Bommel, 2004). For instance, normalizing a conceptual schema, producing an SQL database or XML structures from a conceptual schema, optimizing a physical schema, or reverse engineering standard files and CODASYL structures can be described mostly as schema transformations. In van Bommel (2005), the first author has shown that the whole database design process, together with other related activities, can be described as a chain of schema transformations. Schema transformations are essential to formally define forward and backward mappings between schemas and particularly between conceptual structures and DBMS constructs, which ensures the traceability of the database engineering processes.

Figure 2. The main screen of the Schema Analyzer of DB-MAIN. The script shown is a fragment of the Oracle model.



DB-MAIN includes a tool set of about 30 basic schema transformations, together with facilities to build higher-level transformations. The basic transformations have been selected in order to cover the needs of most database engineering activities, including those mentioned above. In particular, they provide operators to materialize IS-A relations, to transform entity types

Figure 3. A typical schema transformation that replaces a many-to-many relationship type with an entity type and one-to-many relationship types



into relationship types or into attributes, and to split/merge entity types. Attributes can be transformed into entity types and relationship types. Compound attributes can be disaggregated, while multivalued attributes can also be replaced with serial or concatenated attributes. Non-set attributes, such as bags, lists, and arrays, can be transformed into pure sets. Most of them are semantics-preserving, i.e., both source and resulting schemas describe exactly the same information (see Figure 3).

DB-MAIN introduces the concept of *predicate-driven transformation*, noted T(p), where T is any basic transformation and p a structural predicate as described in the previous section. Let us consider:

- RT\_into\_ET that denotes the transformation of a relationship type into an entity type (see Figure 3);
- expression ATT\_per\_RT(i j), a predicate that, when applied to a relationship type, checks whether the number of its attributes is between i and j; and
- expression ROLE\_per\_RT(i j), a predicate that, when applied to a relationship type, checks whether the number of its roles is between i and j.

Then, the expression

RT\_into\_ET(ATT\_per\_RT(1 N) or ROLE\_per\_RT(3 N))

defines a predicate-driven transformation the effect of which, when executed on the current schema, replaces all complex relationship types (i.e., those which have attributes or at least three roles) by entity types.

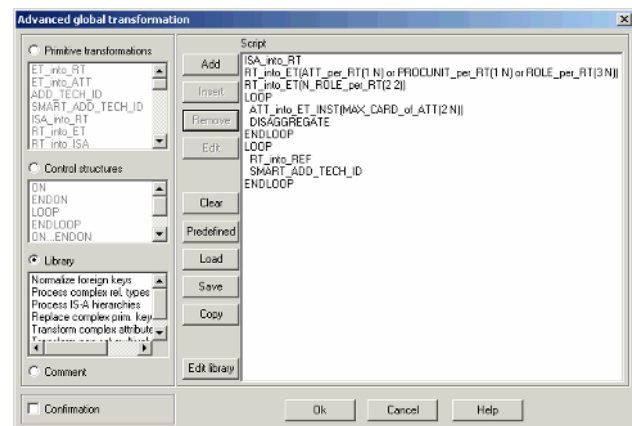
The most powerful application of this concept is the *model-driven transformation*. It consists of a chain of possibly iterative, predicate-driven transformations that is intended to transform any schema into a schema that complies with a definite model.

Both predicate-driven and model-driven transformations can be built, executed, saved, and reused through the Advanced Transformation Assistant of DB-MAIN. A dozen of popular, model-based, predefined transformations are provided to derive relational, XML, and UML schemas, but also to extract conceptual schemas from

DBMS schemas such as CODASYL and standard file managers. Figure 4 shows a fragment of a script that expresses an RDBMS transformation. Though strongly simplified (the complete script comprises about 20 predicate-driven transformations), this fragment includes the most important operators to transform conceptual schemas into relational structures:

- materializing IS-A relations into one-to-one relationship types (ISA\_into\_RT),
- transforming complex relationship types (i.e., with attributes, or with methods, or with at least three roles) into entity types (RT\_into\_ET),
- same for many-to-many relationship types (with two *many* roles),
- transforming multivalued attributes (the maximum cardinality of which is at least two) into entity types (ATT\_into\_ET),
- decomposing compound attributes (DISAGGREGATE),
- applying (LOOP) the latter two until everything has been transformed,

Figure 4. The main screen of the Schema Transformation Assistant of DB-MAIN; the script shown is a fragment of the ERA-to-relational model transformation





- expressing remaining relationship types into referential attributes, i.e., foreign keys (RT\_into\_REF),
- where the latter fails, adding a technical primary key to the concerned entity type (SMART\_ADD\_TECH\_ID), and
- applying (LOOP) the latter two until everything has been transformed.

### Code Generation

A schema that complies with the physical model of a DBMS can be transformed into a DDL script, the execution of which will create the database structures. All CASE tools include generators for a wide list of SQL DBMSs. Theoretically, this should be straightforward since basic generators are fairly easy to develop and since most DBMSs are SQL2 compliant. The hard problem is to translate all the constructs of the conceptual schema into a poor DDL. This means not only that the logical and physical schemas express these constructs, but also that the DDL generator is able to compensate for the weaknesses of the target DDL. For instance, IS-A hierarchies are not correctly translated by most popular SQL generators, and many useful constraints are ignored in the generation process. Consequently, the developer has to manually modify the DDL code to translate the discarded constructs, thus breaking the traceability chain between the conceptual schema and the SQL code.

DB-MAIN offers a collection of basic generators for SQL2 (including Oracle, Access, MySQL, and InterBase), COBOL files, CODASYL (IDS2), and XML (DTD and Schema). It also includes a parametric SQL generator that takes in charge all the IS-A patterns as well as a large set of integrity constraints. The constructs that cannot be explicitly declared in SQL will be controlled through check predicates, views with check option, triggers, or stored procedures.

Some CASE tools also provide generators for DB-related code such as J2EE or .NET components, wrappers, migrators, active components for special-purpose databases, or GUI to databases.

### Reverse Engineering

The main goal of this process is to rebuild the logical and conceptual schemas of a legacy database, the documentation of which is missing or outdated. Since it has been recognized that the availability of these schemas is an absolute prerequisite in database maintenance, evolution, migration, or federation, reverse engineering has gained increasing interest during the last decade. Most CASE tools are able to extract a graphical physical schema from the SQL code or from system catalog tables. They

also provide some elementary transformations to produce a basic conceptual schema, mainly by replacing foreign keys by relationship types. However, such an approach falls short for most actual databases. Indeed, it is based on a oversimplistic hypothesis according to which the conceptual schema has been completely expressed through explicitly declared SQL constructs, and, conversely, the conceptual schema only includes constructs that can be declared through SQL statements.

Unfortunately, most existing databases do not meet this requirement. On the one hand, they often use legacy technologies, such as RPG, CODASYL, IMS, IMAGE, or COBOL files, for which few robust extractors have been developed. On the other hand, many constructs and constraints have not been translated into DDL code, but rather coded in the application programs, notably in data validation sections scattered through millions of lines of code. Discovering them involves in-depth procedural code analysis that is out of the scope of standard text and character stream analyzers. Finally, many database structures are not a straightforward translation of a clean conceptual schema, but include complex optimization patterns that often are difficult to identify and interpret.

DB-MAIN (Hainaut, Englebert, Henrard, Hick & Rolans, 1996) includes a powerful tool set for large database reverse engineering. In particular it offers DDL code extractors for SQL, RPG, IMS, CODASYL (IDMS and IDS2), COBOL files, and XML; programmable text analyzers to identify programming clichés; dependency and dataflow graph analyzers; and a sophisticated program slicer to identify program sections that process database components. It also includes specific assistants to discover implicit foreign keys and scripts to analyze large schemas and to extract conceptual structures from logical schemas. The extensibility services of DB-MAIN make it possible to develop additional tools to cope with new problems such as unknown legacy technologies.

### Meta-CASEs

Strictly speaking, a meta-CASE is an environment in which one can describe and generate customized CASE tools for a specific engineering area (Englebert & Hainaut, 1999; Kelly, Lyttinen, & Rossi, 1996). However, most CASE tools offer some restricted capabilities for developing additional specific, user-defined functions.

DB-MAIN provides developers and method engineers with five meta-level extensibility services.

1. Method fragments (models and model-based transformations) can be developed via the scripting facility of the Schema Analysis and Global Transformation assistants.

2. The repository can be extended by associating user-defined meta-properties with all the object types.
3. New processors and functions can be developed either in Java or through the 4GL Voyager 2.
4. After and before triggers can be defined for all the repository modification and transformation primitives.
5. MDL, a method engineering environment, allows companies to develop specific methodologies (Roland, Hainaut, Hick, Henrard, & Englebert, 2000). A method comprises specific models, product types, and engineering processes defined by their strategies. The method engine of DB-MAIN guarantees the strict application of the chosen methodology. It also ensures complete traceability thanks to its history manager.

For instance, customized versions of DB-MAIN have been developed for data warehouse design, database evolution, database reverse engineering, active database generation, XML engineering, federated database development and generation, and temporal database design and generation.

## FUTURE TRENDS

Despite much effort from laboratories and industry (ISDOS, one of the first CASE tools was developed by Teichrow and Hershey in 1977), CASE tools are permeating more slowly than expected among practitioners. In their in-depth analysis, Lundell and Lings (2004) identify three factors that can explain this phenomenon but fail to spot the poor quality of the artefacts and code that CASE tools produce from users' specifications.

Future tools are expected to provide better code generators; tools interoperability, notably through the CDIF (ISO, 2000); and XMI standards, intelligent assistance, and support for maintenance and evolution. Inter-model synchronization and transformational approaches will be better supported, as witnessed by the increasing popularity of the OMG MDA.

## CONCLUSION

Any attempt to write a stable state of the art in database CASE technology would be hopeless. Indeed, this domain is highly volatile (Lundell & Lings, 2004) and is the subject of rapid merging and buying, so that even the name of well-established CASE tools can change in a short time.

Nevertheless, we can mention, among the main tools at the present time, *Rose* and *XDE* from Rational, *Together* from Borland, *PowerDesigner* from Sybase, *ERwin* from Computer Associates, *MEGA Database* from MEGA International, *MetaEdit+*, a meta-CASE from MetaCase, *Designer/2000* and *Developer/2000* from Oracle, and *Silverrun* from Magna Solutions. Interested readers can also consult the site <http://www.qucis.queensu.ca/Software-Engineering/tools.html>. Information on DB-MAIN can be found at <http://www.info.fundp.ac.be/libd>. The free education edition can also be downloaded from this address.

## REFERENCES

- Batini, C., Ceri, S., & Navathe, S. B. (1992). *Conceptual database design: An entity-relationship approach*. Redwood City, CA: Benjamin Cummings.
- Englebert, V., & Hainaut, J.-L. (1999). DB-MAIN: A next generation meta-CASE. *Information Systems Journal*, 24(2).
- Hainaut, J.-L. (1996). Specification preservation in schema transformations—Application to semantics and statistics. *Data & Knowledge Engineering*, 11(1).
- Hainaut, J.-L. (2005). *Transformation-based database engineering*.
- Hainaut, J.-L., Englebert, V., Henrard, J., Hick J.-M., & Roland, D. (1996). Database reverse engineering: From requirements to CASE tools. *Journal of Automated Software Engineering*, 3(1).
- ISO. (2000). *Information technology—CDIF framework—Part 1: Overview* (ISO/IEC FDIS 15474-1, Final Draft, ISO/IEC JTC 1/SC7/WG11, 2000-02-14).
- Kelly, S., Lyytinen, K., & Rossi, M. (1996). MetaEdit+: A fully configurable multi-user and multi-tool CASE and CAME environment. In P. Constantopoulos, J. Mylopoulos, & Y. Vassiliou (Eds.), *Lecture notes in computer science: Vol. 1080. Proceedings of CAiSE'96*. Berlin/Heidelberg, Germany: Springer-Verlag.
- Lundell, B., & Lings, L. (2004). Changing perceptions of CASE technology. *Journal of Systems and Software*, 72(2).
- Object Management Group. (2003, May). *XML Metadata Interchange (XMI), v2.0*. Retrieved September 2004 from <http://www.omg.org/cgi-bin/doc?formal/2003-05-02>
- Roland, D., Hainaut, J.-L., Hick, J.-M., Henrard, J., & Englebert, V. (2000). Database engineering processes

with DB-MAIN. In *Proceedings of the eighth European Conference on Information Systems (ECIS2000)*, Vienna University, Austria (Vol. 2, pp. 244-251).

Rosenthal, A., & Reiner, D. (1994). Tools and transformations—rigorous and otherwise—for practical database design. *ACM TODS*, 19(2).

Teichrow, D., & Hershey, E. (1977). PSL/PSA: A computer aided technique for structured documentation and analysis of information processing systems. *IEEE TOSE*, 3(1).

van Bommel, P. (Ed.). (2004). *Transformation of knowledge, information and data: Theory and applications*. Hershey, PA: Infosci.

## KEY TERMS

**CASE Tool:** Software tools that help software designers and developers specify, generate, and maintain some or all software components of an application. Most CASE tools provide functions to allow developers to draw database schemas and to generate the corresponding DDL code.

**Conceptual Schema:** A structured technology-independent description of the information about an application domain such as a company or a library. By extension, it is also an abstract representation of the existing or project database that is made up of the data of this domain.

**Database Reverse Engineering:** The process through which the logical and conceptual schemas of a legacy database or of a set of files are recovered or rebuilt from various information sources such as DDL code, data dictionary contents, database contents, or the source code of application programs that use the database.

**DB-MAIN:** An experimental CASE tool in development at the University of Namur since 1993. It supports most database engineering processes, among them, information analysis, database design, reverse engineering, federated database design, and database migration. Its generic structure model and method engineering environment allow users to build their own methodologies.

**Logical Schema:** The description of the data structures of a database according to the model of a specific technology, e.g., a RDBMS. The logical schema of a database is the implementation of its conceptual schema. Application programs know the database through its logical schema.

**Models and Schemas:** In the database realm, a model  $M$  is a formal system comprising a closed set of abstract object categories and a set of assembly rules that states which arrangements of objects are valid. Since  $M$  is supposed to describe the structure, the properties, and the behaviour of a class  $S$  of external systems, the semantics of  $M$  is specified by a mapping of  $M$  onto  $S$ . Any arrangement  $m$  of objects which is valid according to  $M$  describes a specific system  $s$  of class  $S$ .  $m$  is called a schema while  $s$  is the application domain or the universe of discourse. Among the most popular conceptual models we can mention the entity-relationship, object-role, relational, and UML class models. Among DBMS models, the SQL, CODASYL, IMS, object-relational, and XML models are currently the most widely used.

**Physical Schema:** The technical description of a database; all the physical constructs (such as indexes) and parameters (such as page size or buffer management policy) are specified. The physical schema of a database is the implementation of its logical schema.

**Repository:** A database in which the specification and the description of the various artefacts of a software system are stored. As far as database engineering is concerned, the repository includes the schemas of the project database at all the abstraction levels as well as the correspondence between them. All CASE tools rely on some sort of repository.

**Traceability:** The property of software design and development that makes it possible to link any abstract artefact to the technical artefacts that implement it and conversely. In addition, this link explains how and why this implementation has been chosen. In the database realm, traceability allows a programmer to know exactly which conceptual object a definite column is an implementation of. Conversely, it informs him/her on how a conceptual relationship type has been implemented. The transformational paradigm is one of the most promising approaches to formally guarantee traceability.

# Checking Integrity Constraints in a Distributed Database

Hamidah Ibrahim

Universiti Putra Malaysia, Malaysia

## INTRODUCTION

Preserving the accuracy and the integrity of information in a database is extremely important for the organization that is maintaining that database. Such an organization is likely to rely heavily upon that accuracy. Applications that consult and use the database expect a guarantee that the database is supplying the correct information. Critical business decisions may be made assuming that information extracted from the database is correct. Thus, incorrect data can lead to incorrect business decisions, which can have serious implications for the people and organizations using it (Codd, 1990).

An important problem for a database system is to guarantee *database consistency*. Many techniques and tools have been devised to fulfill this requirement in many interrelated research areas, such as concurrency control, security control, reliability control and integrity control (Eswaran & Chamberlin, 1975; Grefen, 1993). Concurrency control deals with prevention of inconsistencies caused by concurrent access by multiple users or applications to a database. Security control deals with preventing users from accessing and modifying data in a database in unauthorized ways. Reliability control deals with the prevention errors due to the malfunctioning of system hardware or software. Integrity control deals with the prevention of semantic errors made by the users due to their carelessness or lack of knowledge.

A database state is said to be consistent if the database satisfies a set of statements, called *semantic integrity constraints* (or simply *constraints*). Integrity constraints specify those configurations of the data that are considered semantically correct. Any update operation (insert, delete, or modify) or transaction (sequence of updates) that occurs must not result in a state that violates these constraints. Thus, a fundamental issue concerning integrity constraints is *constraint checking*, that is the process of ensuring that the integrity constraints are satisfied by the database after it has been updated. Checking the consistency of a database state will generally involve the execution of *integrity tests* on the database which verify whether the database is satisfying its constraints or not.

In a database system, a semantic integrity subsystem (SIS) is responsible for managing and enforcing integrity constraints to ensure that these rules are not violated by the database and the database is in a consistent state. An early proposal by Eswaran and Chamberlin (1975) described the functionality requirements for an integrity subsystem. The main tasks of this subsystem are to determine which constraints are to be checked after each database change and to trigger the appropriate actions when a constraint violation is detected. The crucial problem encountered in designing a complete integrity subsystem is the difficulty of devising an efficient algorithm for enforcing database integrity when updates occur. Many proposals for designing an integrity subsystem can be found in the database literature. In Grefen (1993) and McCarroll (1995) three ways to couple the integrity subsystem to the database management system (DBMS) are described.

The first approach is known as the *decoupled subsystem approach*. This adds the subsystem as an additional layer on top of an existing DBMS. It was employed by the AIM project (Cremers & Domann, 1983) and the KBDTA system (Wang, 1992). In this approach, the responsibility for ensuring the consistency of the database when a transaction occurs is part of the transaction design process. The transaction designers are responsible for ensuring that the transactions are *safe* (i.e., when executed, the transactions are guaranteed to bring the database from one consistent state to another consistent state). Consequently, as transactions can get complex, a *transaction design tool* is usually incorporated into the subsystem to assist the transaction designers to construct safe transactions. Hence, in this approach, little or no support within the DBMS is needed for automatically enforcing database integrity constraints.

The second approach is known as the *loosely coupled subsystem*. It adds the subsystem as an extension to the DBMS. This is employed by the SABRE (Simon & Valduriez, 1987) and the PRISMA (Grefen, 1990) projects. In this approach, transactions have integrity tests embedded in them to perform the necessary integrity checking. The modified transactions can then be executed by the standard transaction facilities. This



approach is based on *query modification* and *transaction modification* strategies, where an arbitrary query or transaction that may violate the integrity of a database is modified, such that the execution of the modified query or transaction is assured to leave the database in a consistent state.

In the third approach, which is known as the *tightly coupled subsystem*, the subsystem is seen as part of the basic functionality of a database system and is fully integrated into it. This approach, initially proposed by Hammer and McLeod (1975) and Eswaran and Chamberlin (1975), is employed by the Starbust project (Ceri, Fraternali, Paraboschi, & Tanca, 1994), SICSDD project (Ibrahim, Gray, & Fiddian, 1998) and the latest versions of commercial DBMSs, such as INGRES and ORACLE. In this approach, integrity tests are general rather than transaction specific and thus no knowledge of the internal structure of a transaction is required. Typically, this requires rule mechanisms to implement integrity constraint enforcement (Ibrahim, 2002a).

## BACKGROUND

The growing complexity of modern database applications plus the need to support multiple users has further increased the need for a powerful integrity subsystem to be incorporated into these systems. Therefore, a complete integrity subsystem is considered to be an important part of any modern DBMS (Grefen, 1993). The crucial problem in designing a complete integrity subsystem is the difficulty of devising an efficient algorithm for enforcing database integrity against updates. Thus, it is not surprising that much attention has been paid to the maintenance of integrity in centralized databases over the last decade. A naive approach is to perform the update and then check whether the integrity constraints are satisfied in the new database state. This method, termed *brute force checking*, is very expensive, impractical and can lead to prohibitive processing costs (Embury, Gray, & Bassiliades, 1993; Hsu & Imielinski, 1985; Mazumdar, 1993, Plexousakis, 1993; Qian, 1988, 1989; Sheard & Stemple, 1989). Enforcement is costly because the evaluation of integrity constraints requires accessing large amounts of data which are not involved in the database update transition (Simon & Valduriez, 1987). Hence, improvements to this approach have been reported in many research papers (Bernstein & Blaustein, 1981; Blaustein, 1981; Henschen, McCune, & Naqvi, 1984; Hsu & Imielinski, 1985; McCune & Henschen, 1989; Nicolas, 1982; Qian, 1989). Although this research effort has yielded fruitful results that have given centralized systems a substantial level of reliability and robustness with respect to the

integrity of their data, there has so far been little research carried out on integrity issues for distributed databases. The problem of devising an efficient enforcement mechanism is more crucial in a distributed environment. This is due to the following facts (Barbara & Garcia-Molina, 1992; Mazumdar, 1993; Qian 1989; Simon & Valduriez, 1987):

- Integrity constraints are general statements about sets of data elements which may spread over several sites in a distributed database. A large amount of data may therefore need to be transferred around the network in order to determine the truth of such statements.
- Owing to the possibility of fragmentation of relations with the fragments stored at different locations, the integrity constraints must be transformed into constraints on the fragments so that they can be straightforwardly used for constructing efficient enforcement algorithms. Thus, there are usually more integrity constraints in an equivalent distributed database than a centralized database, all of which need to be maintained. In addition, replication of data imposes an additional constraint that the replicas must have equivalent values at all times.
- Frequent updates can lead to frequent executions of expensive violation testing operations.
- If some constraints are violated, the whole update transaction which causes the state transition must be aborted and the database must be restored to the previous state, which can be a very costly operation in a distributed system.

The brute force strategy of checking constraints is worse in the distributed context since the checking would typically require data transfer as well as computation leading to complex algorithms to determine the most efficient approach. Allowing an update to execute with the intention of aborting it at commit time in the event of constraint violation is also inefficient since rollback and recovery must occur at all sites which participated in the update. Thus, the question of interest is *how to efficiently check integrity constraints in a distributed environment*.

Many researchers have studied the problem of maintaining the consistency of a database and not surprisingly many different approaches have been proposed. For centralized databases, researchers have suggested that constraint checking can be optimized by exploiting the fact that the constraints are known to be satisfied prior to an update, and by *reducing the number of integrity constraints that need checking* by only checking the subset of integrity constraints that may be vio-



lated by the current update or transaction. This is based on the following observation by Nicolas (1982). Given a valid database state, a new state is obtained when it is updated either by a single update operation or by a transaction. Depending on the operation leading to the new state, some integrity constraints remain necessarily satisfied in this new state, while others have to be evaluated to determine whether they are actually satisfied or not (Nicolas, 1982). Thus an integrity testing strategy which avoids redundantly checking constraints that are satisfied in the database before and are not affected by the update operation is better (more efficient) than a basic strategy which checks all the constraints. This revised strategy, known as *incremental integrity checking* (Gupta, 1994, Plexousakis, 1993) or constraint filtering (Grefen, 1990), is the basis of most current approaches to integrity checking in databases. In Gupta (1994), this strategy is also referred to as a brute force strategy because by default it uses all the underlying relations.

Another strategy is to simplify the constraint formulae so that *less data are accessed* in order to determine the truth of the constraint. With the assumption that the set of initial constraints (*IC*), is known to be satisfied in the state before an update, simplified forms of *IC*, say *IC'*, are constructed such that *IC* is satisfied in the new state if and only if *IC'* is satisfied, and the evaluation cost of *IC'* is less than or equal to the evaluation cost of *IC*. This strategy is referred to as *constraint simplification* and the simplified forms of these constraints are referred to as *integrity tests* (Ibrahim, 2002b; Ibrahim, Gray & Fiddian, 1996; McCarroll, 1995) or *constraint protectors* (Stemple, Mazumdar, & Sheard, 1987). This approach conforms to the admonition of Nicolas (1982) to concentrate on the problem of finding *good* constraints. Various simplification techniques have been proposed where integrity tests are derived from the syntactic structure of the constraints and the update operations (Bernstein & Blaustein, 1981; Blaustein, 1981; Gupta, 1994; Henschen, McCune, & Naqvi, 1984; Hsu & Imielinski, 1985, McCarroll, 1995, McCune & Henschen, 1989; Nicolas, 1982; Qian, 1989; Simon & Valduriez, 1987). These techniques are referred to as *constraint simplification by update analysis*. Researchers in this area have focused solely on the derivation of efficient integrity tests, claiming that they are cheaper to enforce and reduce the amount of data accessed, thus reducing the cost of integrity constraint checking. Three different types of integrity test are defined in McCune and Henschen (1989), namely, *sufficient tests*, *necessary tests*, and *complete tests*. An important property desired of an integrity test of any of these three types is that the test will be cheaper to execute than the initial constraint from which it is derived. Thus, it is important to ensure that such integrity tests are as efficient as possible in

order to reduce the performance overheads imposed by integrity enforcement. The issue addressed here is *how to derive an efficient set of tests to prove that an update operation will guarantee the semantic integrity of the database with respect to each and every constraint defined on the database*.

Furthermore, to avoid undoing the updates, these tests must be evaluated before the database state transition caused by the update occurs. The introduction of inconsistencies in the database is therefore prevented by committing only those update operations that result in a consistent database state. Methods following this approach are term *preventive methods* and are favoured over *detective methods*, which allow an update operation to be executed on a database state and when an inconsistent result state is detected undo this update.

For distributed database environments, most of the strategies proposed for centralized systems are used, in particular the incremental integrity checking strategy and the constraint simplification method strategy, which aim to reduce the number of integrity constraints to be evaluated and the amount of data accessed. In addition, new strategies appropriate to a distributed environment have been proposed which aim to reduce the number of sites involved and thus reduce the amount of data transferred across the network (Barbara & Garcia-Molina, 1992; Gupta, 1994; Ibrahim, Gray & Fiddian, 1998; Mazumdar, 1993; Qian, 1989). These strategies try to avoid remote accesses to the update sites (target sites) and are invaluable in a situation where it is impossible to access data located at other sites in the distributed system, for example in situations where network failure is detected, or a high-security database is involved. This means identifying *how to simplify the integrity constraints so that evaluation is more local and at best is entirely local to the sites involved in the update*.

Although the performance of constraint checking in a distributed database system has been improved by adopting the centralized strategies, with respect to the amount of data accessed or transferred, the following strategies are still inefficient for distributed environments:

- i. Most of the simplified forms are derived from the initial constraints as specified by the user. These derivations do not exploit knowledge about the database application, especially its data fragmentation and allocation, which can be used to
  - a. derive a set of simplified constraints that can be straightforwardly used for constructing efficient enforcement algorithms with respect to the distributed environment, and

- b. infer the information stored at different sites of the network and so minimize the support from remote sites required when checking the constraints.
- ii. Complete tests, which are the tests used most often in centralized systems, are usually expensive in a distributed environment. However, sufficient tests are useful in a distributed environment (Mazumdar, 1993) because their checking space often only spans a single site and therefore they can be performed at a reasonable cost as the number of sites involved and the amount of data transferred across the network are reduced.

The problem of choosing a good set of constraints for better enforcement that has been explored intensively in centralized systems has been relatively neglected for distributed systems.

Constraint simplification methods in a distributed environment can be classified into two types of approach. Both approaches are targeted at deriving integrity tests/conditions, but they use different information. The first approach uses knowledge about the application domain and data fragmentation (Mazumdar, 1993; Qian, 1989). This approach is referred to as *constraint simplification by reformulation*. The second approach analyses the syntactic structure of the constraints and the update operations in order to generate the integrity tests, as reported in Gupta (1994), and is referred to as *constraint simplification by update analysis*.

## STRATEGIES FOR CHECKING INTEGRITY CONSTRAINTS IN A DISTRIBUTED DATABASE

In order to improve and to generate an efficient integrity constraint checking in a distributed environment the following circumstances should be undertaken.

- i. **Constraint Filtering:** It assumes that the database is consistent prior to an update and an incremental integrity checking strategy is adopted which *reduces the number of constraints evaluated*. For each update request, only those constraints that may be violated by it are selected for further evaluation. This can be performed by observing the following rules where integrity constraints are specified in prenex conjunctive normal form.
  - a. Whenever an update operation is dealing with the extension of a relation  $R$ , integrity con-

straints in which  $R$  does not occur are unaffected.

- b. Integrity constraints which do not contain  $R$  in negated atomic formula are unaffected when a tuple is inserted into the extension of  $R$ .
- c. Integrity constraints which do not contain  $R$  in a nonnegated atomic formula are unaffected when a tuple is deleted from the extension of  $R$ .

Also, by rescheduling the execution of the integrity rules, early update abortion in the case of constraint violation is effected. Detecting constraint violation as early as possible is important since it eliminates the execution of further integrity rules and thus reduces the execution time of the process. Some heuristic rules can be applied, such as the following:

- a. Choose an integrity constraint (rule) whose violation implies that no other integrity constraints are triggered, i.e. an isolated node in the triggering graph. For example, an integrity rule for a key constraint with an ABORT action. This is because inserting a new tuple which violates a key constraint implies that no insertion should be made at all.
  - b. Choose a local rule, i.e. an integrity rule that can be performed at a local site.
  - c. An integrity rule with test  $T_i$  which subsumes another integrity rule with test  $T_j$  is preferred since the truth of test  $T_i$  implies the truth of test  $T_j$ .
- ii. **Constraint Optimization:** Because the enforcement of the constraints takes place at the fragment level, the constraints specified in terms of global relations should be transformed into constraints specified in terms of fragment relations so that they can be straightforwardly used for constructing efficient enforcement algorithms. Here, efficient means a set of fragment constraints, which is semantically equivalent to the initial set, does not contain any semantic or syntactically redundant fragment constraints/constructs, eliminates any fragment constraints whose evaluation is proven to be true, eliminates any fragment constraints which contradict already existing fragmentation rules and whose derived constraints are either more local (less distributed) or are entirely local when compared with the initial set. Thus, the properties that one should look for in the derived fragment constraints are that they are *more efficient* and *more local* than the initial set of constraints. Most of the previous approaches/methods proposed for finding/deriving a good set of constraints concentrate on deriving

simplified forms of the constraints by analyzing both the syntax of the constraints and their appropriate update operations. These methods are based on syntactic criteria. In fact, an improved set of constraints (fragment constraints) can be constructed by applying both the semantic and syntactic methods. Also, the knowledge about the data distribution and the application domain can be used to create algorithms which derive using this knowledge an efficient set of fragment constraints from the original constraints. In order to derive an integrity tests from a given initial constraint, the following steps can be followed (Ibrahim, Gray & Fiddian, 2001):

- a. Transform the initial constraint into a set of logically equivalent fragment constraints which reflect the data fragmentation. At this stage, the transformations are restricted to logically equivalent transformations, without considering any reformulation of the original constraints. The transformation is one to many, which means that given an integrity constraint, the result of the transformation is a logically equivalent set of fragment constraints. There are six transformation rules that are applied during this process (Ibrahim, Gray & Fiddian, 2001) which cover the horizontal, vertical and mixed fragmentation.
  - b. Optimize the derived set of fragment constraints. There are several types of optimization technique that can be applied to constraints at the fragment level, such as techniques for optimizing query processing (Chakravarthy, 1990), reformulation techniques (Qian, 1989) and theorem based techniques (McCarroll, 1995; McCune & Henschen, 1989). Constraint optimization can be performed before (preoptimization) or after (postoptimization), compilation. The constraint optimization process should use both syntactic and semantic information about the database integrity constraints.
  - c. Distribute the set of fragment constraints to the appropriate site(s). Because the complexity of enforcing constraints is directly related to both the number of constraints in the constraint set and the number of sites involved, the objective of this phase is to reduce the number of constraints allocated to each site for execution at that site. Distributing the whole set of fragment constraints to every site is not cost-effective since not all fragment constraints are affected by an update and so sites may not be affected by particular updates. The decision of the distribution is based on the site allocation of the fragment relations specified in the constraint.
  - d. Generate the integrity tests. There are a lot of techniques proposed by previous researchers to generate integrity tests (simplified forms) as discussed in the previous section.
- These four steps use the assumption that the database is consistent prior to an update operation, the fragmentation strategies used, the allocation of the fragment relations, the specification of the fragment constraints, and the generated update templates.
- iii. **Localizing Integrity Checking:** It assumes efficient integrity tests can be generated. Here, efficiency is measured by analyzing three components, namely, *the amount of data that needs to be accessed, the amount of data that needs to be transferred across the network and the number of sites that are involved* in order to check a constraint. The intention is to derive local tests which are complete, necessary or sufficient and whose evaluation is local to a site. Constraint simplification by update analysis is preferred to constraint simplification by reformulation, since reformulation will generally require an unbounded search over all possible constraints when generating the integrity tests. Using heuristics to minimize this search may mean some optimal reformulations are not found. The fragment constraints derived by the transformation process can be classified as either local or nonlocal fragment constraints. The evaluation of a local fragment constraint is performed at a single site, thus it is similar to the evaluation of a constraint in a centralized system. Therefore, the constraint simplification techniques proposed by previous researchers can be adopted for constructing integrity tests for local fragment constraints. For nonlocal fragment constraints, a process which derives local tests from these constraints is more attractive than one which derives global tests whose evaluation spans more than one site. This means exploiting the relevant general knowledge to derive local tests. This will involve using, for example, techniques that can infer the information stored at different sites of the network.
  - iv. **Pretest Evaluation:** It adopts an evaluation scheme which avoids the need to undo the update in the case of constraint violation, i.e. pre-test evaluation is preferred as it avoids the undoing process.
  - v. **Test Optimization:** Integrity test evaluation costs can be further reduced by examining the semantics of both the tests and the relevant update operations. An integrity test for a modify operation can be simplified if the attribute(s) being modified is not the attribute(s) being tested. Also, in some

cases it is more efficient to construct transition tests. These simplified tests further reduce the amount of data needing to be accessed or the number of sites that might be involved, and they are more selective as more constants are being substituted for the variables in these tests, which make them easier and cheaper to evaluate than the generated initial tests.

## FUTURE TRENDS

From the above sections, it is obvious that a constraint checking mechanism for a distributed database is said to be efficient if it can minimize the number of sites involved, the amount of data accessed and the amount of data transferred across the network during the process of checking and detecting constraint violation. The overall intention of a constraint checking mechanism is shown in Figure 1. Three main components can be identified with respect to a given integrity constraint which affect the efficiency of its evaluation. These components which are shown in Figure 1 are represented by the following: (i) The X-axis represents the number of sites involved,  $\sigma$ , in verifying the truth of the integrity constraint; (ii) the Y-axis represents the amount of data accessed (or the checking space),  $A$ ; (iii) the Z-axis represents the amount of data transferred across the network,  $T = \sum_{i=1}^n dt_i$  where  $dt_i$  is the amount of data transferred from site  $i$ , and  $n$  is the number of remote sites involved in verifying the truth of the integrity constraint.

A constraint checking mechanism is said to be more efficient if it tends towards the origin, (0, 0, 0) in this diagram. The strategy should try to allocate the responsibility of verifying the consistency of a database to a single site, i.e. the site where the update operation is to be performed. Thus the number of sites involved,  $\sigma = 1$ . As the checking operation is carried out at a single site, no

transferring of data across the network is required (Gupta, 1994), so  $T = 0$ . The evaluation of local tests involves a checking space and an amount of data accessed (assumed  $A = a_i$ ) which are always smaller than the checking space and the amount of data accessed by the initial constraints since these tests are the simplified forms of those constraints so these are minimized as well.

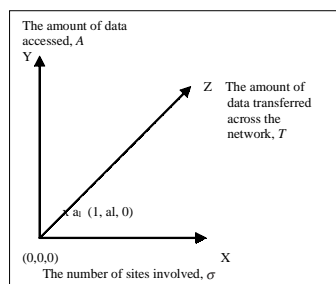
## CONCLUSION

An important aim of a database system is to guarantee database consistency, which means that the data contained in a database is both accurate and valid. There are many ways which inaccurate data may occur in a database. Several issues have been highlighted with regards to checking integrity constraints. The main issue is *how to efficiently check integrity constraints in a distributed environment*. Several strategies can be applied to achieve efficient constraint checking in a distributed database such as constraint filtering, constraint optimization, localizing constraint checking, pre-test evaluation and test optimization. Here, efficiency is measured by analyzing three components, namely: *the amount of data that needs to be accessed, the amount of data that needs to be transferred across the network and the number of sites that are involved* in order to check a constraint.

## REFERENCES

- Barbara, D., & Garcia-Molina, II (1992). The demarcation protocol: A technique for maintaining linear arithmetic constraints in distributed database systems. *Proceedings of the Conference on Extending Database Technology (EDBT'92)* (pp. 373-388).
- Bernstein, P. A., & Blaustein, B. T. (1981). A simplification algorithm for integrity assertions and concrete views. *Proceedings of the 5<sup>th</sup> International Computer Software and Applications Conference (COMPSAC'81)* (pp. 90-99).
- Blaustein, B. T. (1981). *Enforcing database assertions: Techniques and applications*. Doctoral dissertation, Harvard University.
- Ceri, S., Fraternali, P., Paraboschi, S., & Tanca, L. (1994). Automatic generation of production rules for integrity maintenance. *ACM Transactions on Database Systems*, 19(3), 367-422.

Figure 1. The intention of the constraint checking mechanism





- Chakravarthy, U. S. (1990). Logic-based approach to semantic query optimization. *ACM Transactions on Database Systems*, 15(2), 162-207.
- Codd, E. F. (1990). *The relational model for database management* (Version 2). Boston: Addison-Wesley.
- Cremers, A. B., & Domann, G. (1983). AIM—An integrity monitor for the database system INGRES. *Proceedings of the 9<sup>th</sup> International Conference on Very Large Data Bases (VLDB 9)* (pp. 167-170).
- Embury, S. M., Gray, P. M. D., & Bassiliades, N. D. (1993). Constraint maintenance using generated methods in the P/FDM object-oriented database. *Proceedings of the 1<sup>st</sup> International Workshop on Rules in Database Systems* (pp. 365-381).
- Eswaran, K. P., & Chamberlin, D. D. (1975). Functional specifications of a subsystem for data base integrity. *Proceedings of the 1<sup>st</sup> International Conference on Very Large Data Bases (VLDB 1)*, 1(1), 48-68.
- Grefen, P. W. P. J. (1990). *Design considerations for integrity constraint handling in PRISMA/DBI* (Prisma Project Document P508). University of Twente, Enschede.
- Grefen, P. W. P. J. (1993). Combining theory and practice in integrity control: A declarative approach to the specification of a transaction modification subsystem. *Proceedings of the 19<sup>th</sup> International Conference on Very Large Data Bases (VLDB 19)* (pp. 581-591).
- Gupta, A. (1994). *Partial information based integrity constraint checking*. Doctoral dissertation, Stanford University.
- Hammer, M. M., & McLeod, D. J. (1975). Semantic integrity in a relational data base system. *Proceedings of the 1<sup>st</sup> International Conference on Very Large Data Bases (VLDB 1)*, 1(1), 25-47.
- Henschen, L. J., McCune, W. W., & Naqvi, S. A. (1984). Compiling constraint-checking programs from first-order formulas. *Advances in Database Theory*, 2, 145-169.
- Hsu, A., & Imielinski, T. (1985). Integrity checking for multiple updates. *Proceedings of the 1985 ACM SIGMOD International Conference on the Management of Data* (pp. 152-168).
- Ibrahim, H. (2002a). Extending transactions with integrity rules for maintaining database integrity. *Proceedings of the International Conference on Information and Knowledge Engineering (IKE'02)* (pp. 341-347).
- Ibrahim, H. (2002b). A strategy for semantic integrity checking in distributed databases. *Proceedings of the 9<sup>th</sup> International Conference on Parallel and Distributed Systems (ICPADS 2002)* (pp. 139-144). IEEE Computer Society.
- Ibrahim, H., Gray, W. A., & Fiddian, N. J. (1996). The development of a semantic integrity constraint subsystem for a distributed database (SICSDD). *Proceedings of the 14<sup>th</sup> British National Conference on Databases (BNCOD 14)* (pp. 74-91).
- Ibrahim, H., Gray, W. A., & Fiddian, N. J. (1998). SICSDD—A semantic integrity constraint subsystem for a distributed database. *Proceedings of the 1998 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '98)* (pp. 1575-1582).
- Ibrahim, H., Gray, W. A., & Fiddian, N. J. (2001). Optimizing fragment constraints—A performance evaluation. *International Journal of Intelligent Systems—Verification and Validation Issues in Databases, Knowledge-Based Systems, and Ontologies*, 16(3), 285-306.
- Mazumdar, S. (1993). Optimizing distributed integrity constraints. *Proceedings of the 3<sup>rd</sup> International Symposium on Database Systems for Advanced Applications* (Vol. 4, pp. 327-334).
- McCarroll, N. F. (1995). *Semantic integrity enforcement in parallel database machines*. Doctoral dissertation, University of Sheffield.
- McCune, W. W., & Henschen, L. J. (1989). Maintaining state constraints in relational databases: A proof theoretic basis. *Journal of the Association for Computing Machinery*, 36(1), 46-68.
- Nicolas, J. M. (1982). Logic for improving integrity checking in relational data bases. *Acta Informatica*, 18(3), 227-253.
- Plexousakis, D. (1993). Integrity constraint and rule maintenance in temporal deductive knowledge bases. *Proceedings of the 19<sup>th</sup> International Conference on Very Large Data Bases (VLDB 19)* (pp. 146-157).
- Qian, X. (1988). An effective method for integrity constraint simplification. *Proceedings of the 4<sup>th</sup> International Conference on Data Engineering (ICDE 88)* (pp. 338-345).
- Qian, X. (1989). Distribution design of integrity constraints. *Proceedings of the 2<sup>nd</sup> International Conference on Expert Database Systems* (pp. 205-226).
- Sheard, T., & Stemple, D. (1989). Automatic verification of database transaction safety. *ACM Transactions on Database Systems*, 14(3), 322-368.



## Checking Integrity Constraints in a Distributed Database

Simon, E., & Valduriez, P. (1987). *Design and analysis of a relational integrity subsystem* (Tech. Rep. DB-015-87). Austin, TX: MCC.

Stemple, D., Mazumdar, S., & Sheard, T. (1987). On the modes and measuring of feedback to transaction designers. *Proceedings of the 1987 ACM-SIGMOD International Conference on the Management of Data* (pp. 374-386).

Wang, X. Y. (1992). *The development of a knowledge-based transaction design assistant*. Doctoral dissertation, University of Wales College of Cardiff.

### KEY TERMS

**Complete Test:** Verifies that an update operation leads a consistent database state to either a consistent or inconsistent database state.

**Data Fragmentation:** Refers to the technique used to split up the global database into logical units. These logical units are called *fragment relations*, or simply *fragments*.

**Database Consistency:** Means that the data contained in the database is both accurate and valid.

**Distributed Database:** A collection of multiple, logically interrelated databases distributed over a computer network.

**Global Test:** Verifies that an update operation violates an integrity constraint by accessing data at remote sites.

**Integrity Control:** Deals with the prevention of semantic errors made by the users due to their carelessness or lack of knowledge.

**Local Test:** Verifies that an update operation violates an integrity constraint by accessing data at the local site.

**Necessary Test:** Verifies that an update operation leads a consistent database state to an inconsistent database state.

**Sufficient Test:** Verifies that an update operation leads a consistent database state to a new consistent database state.

C

# Collective Knowledge Composition in a P2P Network

**Boanerges Aleman-Meza**

*University of Georgia, USA*

**Christian Halaschek-Wiener**

*University of Georgia, USA*

**I. Budak Arpinar**

*University of Georgia, USA*

## INTRODUCTION

Today's data and information management tools enable massive accumulation and storage of knowledge that is produced through scientific advancements, personal and corporate experiences, communications, interactions, and so forth. In addition, the increase in the volume of this data and knowledge continues to accelerate. The willingness and the ability to share and use this information are key factors for realizing the full potential of this knowledge scattered over many distributed computing devices and human beings. By correlating these isolated islands of knowledge, individuals can gain new insights, discover new relations (Sheth, Arpinar & Kashyap, 2003), and produce more knowledge. Despite the abundance of information, knowledge starvation still exists because most of the information cannot be used effectively for decision-making and problem-solving purposes. This is in part due to the lack of easy to use knowledge sharing and collective discovery mechanisms. Thus, there is an emerging need for knowledge tools that will enable users to collectively create, share, browse, and query their knowledge.

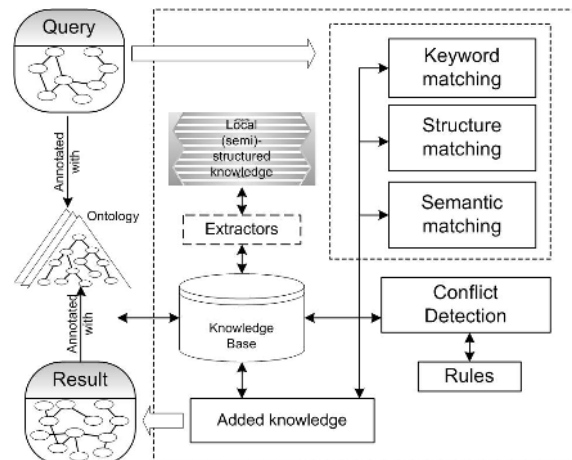
For example, many complex scientific problems increasingly require collaboration between teams of scientists who are distributed across space and time and who belong to diverse disciplines (Loser et al., 2003; Pike et al., 2003). Effective collaboration remains dependent, however, on how individual scientists (i.e., peers) can represent their meaningful knowledge, how they can query and browse each others' knowledge space (knowledge map), and, most importantly, how they can compose their local knowledge pieces together collectively to discover new insights that are not evident to each peer locally.

A common metaphor for knowledge is that it consists of separate little factoids and that these knowledge "atoms" can be collected, stored, and passed along (Lakoff & Johnson, 1983). Views like this are what underlie the notion that an important part of knowledge management is getting access to the "right knowledge."

While the state of the art is not at the point where we can duplicate the accomplishments of a Shakespeare or Einstein on demand, research developments allow us to craft technological and methodological support to increase the creation of new knowledge, both by individuals and by groups (Thomas, Kellogg & Erickson, 2001).

A Peer-to-Peer (P2P) network can facilitate scalable composition of knowledge compared to a centralized architecture where local knowledge maps are extracted and collected in a server periodically to find possible compositions. This kind of vision can be realized by exploiting advances in various fields. Background and enabling technologies include semantic metadata extraction and annotation, and knowledge discovery and composition. Figure 1 shows these components for an ontology-based P2P query subsystem.

Figure 1. Part of an ontology-based P2P query subsystem



## BACKGROUND

### Semantic Metadata Extraction and Annotation

A peer's local knowledge can be in various formats such as Web pages (unstructured), text documents (unstructured), XML (semi-structured), RDF or OWL, and so forth. In the context of efficient collective knowledge composition, this data must be in a machine processable format, such as RDF or OWL. Thus, all data that is not in this format must be processed and converted (metadata extraction). Once this is completed, the knowledge will be suitable to be shared with other peers. The Semantic Web envisions making content machine processable, not just readable or consumable by human beings (Berners-Lee, Hendler & Lassila, 2001). This is accomplished by the use of ontologies which involve agreed terms and their relationships in different domains. Different peers can agree to use a common ontology to annotate their content and/or resolve their differences using ontology mapping techniques. Furthermore, peers' local knowledge will be represented in a machine processable format, with the goal of enabling the automatic composition of knowledge.

Ontology-driven extraction of domain-specific semantic metadata has been a highly researched area. Both semi-automatic (Handschuh, Staab & Studer, 2003) and automatic (Hammond, Sheth & Kochut, 2002) techniques and tools have been developed, and significant work continues in this area (Vargas-Vera et al., 2002).

### Knowledge Discovery and Composition

One of the approaches for knowledge discovery is to consider relations in the Semantic Web that are expressed semantically in languages like RDF(S). Anyanwu and Sheth (2003) have formally defined particular kinds of relations in the Semantic Web, namely, Semantic Associations. Discovery and ranking of these kinds of relations have been addressed in a centralized system (Aleman-Meza et al., 2005; Sheth et al., 2005). However, a P2P approach can be exploited to make the discovery of knowledge more dynamic, flexible, and scalable. Since different peers may have knowledge of related entities and relationships, they can be interconnected in order to provide a solution for a scientific problem and/or to discover new knowledge by means of composing knowledge of the otherwise isolated peers.

In order to exploit peers' knowledge, it is necessary to make use of knowledge query languages. A vast amount of research has been aimed at the development of query languages and mechanisms for a variety of knowledge

representation models. However, there are additional special considerations to be addressed in distributed dynamic systems such as P2P.

## PEER-TO-PEER NETWORKS

Recently, there has been a substantial amount of research in P2P networks. For example, P2P network topology has been an area of much interest. Basic peer networks include random coupling of peers over a transport network such as Gnutella (<http://www.gnutella.com>, discussed by Ripeanu, 2001) and centralized server networks such as that of Napster (<http://www.napster.com>) architecture. These networks suffer from drawbacks such as scalability, lack of search guarantees, and bottlenecks. Yang and Garcia-Molina (2003) discussed super-peer networks that introduce hierarchy into the network in which super-peers have additional capabilities and duties in the network that may include indexing the content of other peers. Queries are broadcasted among super-peers, and these queries are then forwarded to leaf peers. Schlosser et al. (2003) proposed HyperCup, a network in which a deterministic topology is maintained and known of by all nodes in the network. Therefore, nodes at least have an idea of what the network beyond their scope looks like. They can use this globally available information to reach locally optimal decisions while routing and broadcasting search messages. Content addressable networks (CAN) (Ratnasamy et al., 2001) have provided significant improvements for keyword search. If meta-information on a peer's content is available, this information can be used to organize the network in order to route queries more accurately and for more efficient searching. Similarly, ontologies can be used to bootstrap the P2P network organization: peers and the content that they provide can be classified by relating their content to concepts in an ontology or concept hierarchy. The classification determines, to a certain extent, a peer's location in the network. Peers can use their knowledge of this scheme to route and broadcast queries efficiently.

Peer network layouts have also combined multiple ideas briefly mentioned here. In addition, Nejdil et al. (2003) proposed a super-peer based layout for RDF-based P2P networks. Similar to content addressable networks, super-peers index the metadata context that the leaf peers have.

Efficient searching in P2P networks is very important as well. Typically, a P2P node broadcasts a search request to its neighboring peers who propagate the request to their peers and so on. However, this can be dramatically improved. For example, Yang and Garcia-Molina (2003) have described techniques to increase search effectiveness. These include iterative deepening, directed Breadth First Search, and local indices over the data contained



within  $r$ -hops from itself. Ramanathan, Kalogeraki, and Pruyne (2001) proposed a mechanism in which peers monitor which other peers frequently respond successfully to their requests for information. When a peer is known to frequently provide good results, other peers attempt to move closer to it in the network by creating a new connection with that peer. This leads to clusters of peers with similar interests that allow limiting the depth of searches required to find good results. Nejdil et al. (2003) proposed using the semantic indices contained in super-peers to forward queries more efficiently. Yu and Singh (2003) proposed a vector-reputation scheme for query forwarding and reorganization of the network. Tang, Xu, and Dwarkadas (2003) made use of data semantics in the pSearch project. In order to achieve efficient search, they rely on a distributed hash table to extend LSI and VSM algorithms for their use in P2P networks.

## FUTURE TRENDS

Knowledge composition applications are fundamentally based on advances in research areas such as information retrieval, knowledge representation, and databases. As the growth of the Web continues, knowledge composition will likely exploit pieces of knowledge from the multitude of heterogeneous sources of Web content available. The field of peer-to-peer networks is, as of now, an active research area with applicability as a framework for knowledge composition. Given our experiences, we believe that future research outcomes in peer-to-peer knowledge composition will make use of a variety of knowledge sources. Knowledge will be composed from structured data (such as relational databases), semi-structured data (such as XML feeds), semantically annotated data (using the RDF model or OWL), and necessary conversions will be done using knowledge extractors. Thus, knowledge will be composed from databases, XML, ontologies, and extracted data. However, the more valuable insights will probably be possible by combining knowledge sources with unstructured Web content. Large scale analysis and composition of knowledge exploiting massive amounts of Web content remain challenging and interesting topics.

## CONCLUSION

The problem of collectively composing knowledge can greatly benefit from research in the organization and discovery of information in P2P networks. Additionally, several capabilities in creating knowledge bases from heterogeneous sources provide the means for exploiting semantics in data and knowledge for knowledge composition purposes. In this respect, we have discussed the evolution

of peer-to-peer systems from a knowledge composition perspective. Although challenging research problems remain, there is great potential for moving from centralized knowledge discovery systems towards a distributed environment. Thus, research in databases, information retrieval, semantic analytics, and P2P networks provides the basis of a framework in which applications for knowledge composition can be built.

## REFERENCES

- Aleman-Meza, B., Halaschek-Wiener, C., Arpinar, I.B., & Sheth, A. (2005). *Ranking complex relationships on the Semantic Web*.
- Anyanwu, K., & Sheth, A. (2003). r-Queries: Enabling querying for semantic associations on the Semantic Web. *Proceedings of the 12th International World Wide Web Conference*, Budapest, Hungary.
- Berners-Lee, T., Hendler, J., & Lassila, O. (2001). *The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities*. *Scientific American*, 284(5), 34.
- Hammond, B., Sheth, A., & Kochut, K. (2002). Semantic enhancement engine: A modular document enhancement platform for semantic applications over heterogeneous content. In V. Kashyap & L. Shklar (Eds.), *Real world Semantic Web applications* (pp. 29-49). Ios Pr.
- Handschuh, S., Staab, S., & Studer, R. (2003). Leveraging metadata creation for the Semantic Web with CREAM. *KI 2003: Advances in Artificial Intelligence*, 2821, 19-33.
- Lakoff, G., & Johnson, M. (1983). *Metaphors we live by*. Chicago: University of Chicago Press.
- Loser, A., Wolpers, M., Siberski, W., & Nejdil, W. (2003). Efficient data store and discovery in a scientific P2P network. *Proceedings of the ISWC 2003 Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data*, Sanibel Island, Florida.
- Nejdil, W., Wolpers, M., Siberski, W., Schmitz, C., Schlosser, M., Brunkhorst, I., & Löser, A. (2003). Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks. *Proceedings of the 12th International World Wide Web Conference*, Budapest, Hungary.
- Pike, W., Ahlqvist, O., Gahegan, M., & Oswal, S. (2003). Supporting collaborative science through a knowledge and data management portal. *Proceedings of the ISWC 2003 Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data*, Sanibel Island, Florida.



Ramanathan, M.K., Kalogeraki, V., & Pruyne, J. (2001). *Finding good peers in peer-to-peer networks* (Tech. Rep. No. HPL-2001-271). HPLabs.

Ratnasamy, S., Francis, P., Handley, M., Karp, R., & Shenker, S. (2001). A scalable content addressable network. *Proceedings of the ACM SIGCOMM*. New York: ACM Press.

Ripeanu, M. (2001). Peer-to-peer architecture case study: Gnutella Network. *Proceedings of the International Conference on Peer-to-Peer Computing*, Linköping, Sweden.

Schlosser, M., Sintek, M., Decker, S., & Nejd, W. (2003). HyperCuP-Hypercubes, ontologies and efficient search on P2P networks. *Lecture Notes in Artificial Intelligence*, 2530, 112-124.

Sheth, A., Aleman-Meza, B., Arpinar, I.B., Halaschek, C., Ramakrishnan, C., Bertram, C., et al. (2005). Semantic association identification and knowledge discovery for national security applications. In L. Zhou & W. Kim (Eds.), *Special Issue of Journal of Database Management on Database Technology for Enhancing National Security*, 16(1), 33-53. Hershey, PA: Idea Group Publishing.

Sheth, A., Arpinar, I.B., & Kashyap, V. (2003). Relationships at the heart of Semantic Web: Modeling, discovering, and exploiting complex semantic relationships. In M. Nikraves, B. Azvin, R. Yager, & L.A. Zadeh (Eds.), *Enhancing the power of the Internet studies in fuzziness and soft computing* (pp. 63-94). Springer-Verlag.

Tang, C., Xu, Z., & Dwarkadas, S. (2003). Peer-to-peer information retrieval using self-organizing semantic overlay networks. *Proceedings of the ACM SIGCOMM 2003*, Karlsruhe, Germany.

Thomas, J.C., Kellogg, W.A., & Erickson, T. (2001). The knowledge management puzzle: Human and social factors in knowledge management. *IBM Systems Journal*, 40(4), 863-884.

Vargas-Vera, M., Motta, E., Domingue, J., Lanzoni, M., Stutt, A., & Ciravegna, F. (2002). MnM: Ontology driven semi-automatic and automatic support for semantic markup. *Proceedings of the 13th International Conference on Knowledge Engineering and Management*, Sigüenza, Spain.

Yang, B., & Garcia-Molina, H. (2003). Designing a super-peer network. *Proceedings of the 19th International Conference on Data Engineering*, Bangalore, India.

Yu, B., & Singh, M.P. (2003). Searching social networks. *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems*, Melbourne, Australia.

## KEY TERMS

**Knowledge Composition:** Knowledge composition involves assembling knowledge atoms (such as triples in RDF and OWL) to build more complex knowledge maps.

**Metadata:** In general terms, metadata are data about data. Examples are size of a file, topic of a news article, etc and so forth.

**Ontology:** From a practical perspective, ontologies define a vocabulary to describe how *things* are related. Relationships of type “is-a” are very basic, yet taxonomies are built with is-a relationships. The value of ontologies is in the *agreement* they are intended to provide (for humans, and/or machines).

**OWL:** The OWL Web Ontology Language is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics (OWL Web Ontology Language Overview, W3C Recommendation, February 2004).

**RDF(S):** The Resource Description Framework is a language intended for representation and description of ‘resources’. RDF makes use of a Vocabulary Description Language (commonly referred as RDFS or RDF Schema) to describe classes and relations among resources. With RDF(S), we refer to both RDF and its accompanying vocabulary description language. (RDF Primer, W3C Recommendation, February 2004).

**Semantic Metadata:** We refer to ‘semantic metadata’ as that data about data that describes the content of the data. A representative example of semantic metadata is relating data with classes of an ontology, that is, the use of ontology for describing data.

**Semantic Web:** The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners (W3C). The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation (Tim Berners-Lee, James Hendler, Ora Lassila, The Semantic Web, *Scientific American*, May 2001).



# Common Information Model

**James A. Fulton**

*The Boeing Company, USA (Retired)*

## INTRODUCTION

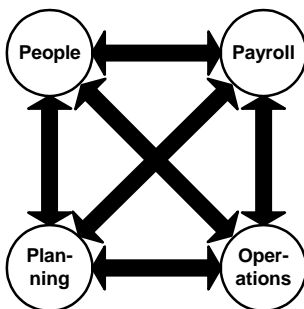
A *common information model (CIM)* defines information that is available for sharing among multiple business processes and the applications that support them. These common definitions are neutral with respect to the processes that produce and use that information, the applications that access the data that express that information, and the technologies in which those applications are implemented. In an architecture based on the CIM, applications map their data only to the CIM and not to other applications, and they interface only to the middleware that implements the CIM (the *integration broker* or *IB*), not to other applications. This not only reduces the number of interfaces to be built and maintained, it provides a basis for integrating applications in a way that reduces the coupling among them, thereby allowing them to be upgraded or replaced with minimal functional impact on other applications.

## BACKGROUND

The concept of a common information model first emerged in public prominence under the name *conceptual schema* with the publication of the ANSI/SPARC Database Model (Jardine, 1977). At the time, it was intended as an approach to designing very large shared database systems.

The key thesis of the ANSI/SPARC Committee was that traditional integration practices required *point-to-point* interfaces between each application and the data sources it depended on, as illustrated in Figure 1. This required the development of a mapping between the

Figure 1. Two-schema architecture



definition of the data available from the source (the *internal schema*) and the definition of the data required by the application (the *external schema*). The committee referred to such an approach as a *two-schema architecture*. The interfaces between the components had to implement this mapping in order to transform the physical representation, the syntax, and the semantics of the data from its source to its destination.

Although such computing systems are relatively simple and quick to build, they impose significant problems during maintenance:

- **Number of Interfaces:** As these systems mature, the number of interfaces to be built and maintained could grow with the square of the number of applications. If  $N$  is the number of applications,  $I$  is the number of interfaces, and each application has one interface in each direction with every other application, then

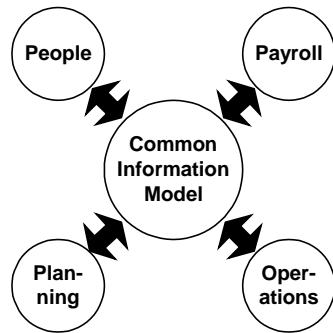
$$I = 2 \times N \times (N-1) = 2N^2 - 2$$

This is known as the *N-Squared Problem*. Although it is rare that every application talks to every other application, many applications share multiple interfaces. So the equation offers a reasonable approximation.

- **Redundancy:** The transformations required to implement communications are implemented redundantly by multiple interfaces. Each application must design its own approach to merging data from multiple sources.
- **Impact Assessment:** When an application changes in a way that affects its internal or external schema, every application that maps to that schema must be examined, reverified, and possibly revised.
- **Scope of Knowledge:** When an application is being upgraded to support new requirements, or when a new application is added to the architecture, architects have to examine every other application to determine what interfaces are required.

The ANSI/SPARC Committee recommended an alternative architecture, as depicted in Figure 2, in which interaction among applications is mediated through an integration broker that implements what they called a

Figure 2. Three-schema architecture



*conceptual schema*, and what this paper refers to as a *common information model* or *CIM* (also known as a *Common Business Information Model*, *Canonical Business Information Model*, *Normalized Information Model*, *Common Business Object Model*). In this architecture, individual applications map their schemas only to the conceptual schema and interface only to the component that implements the conceptual schema (herein referred to as the *integration broker*), which is responsible for translating data from the source application first into the neutral form of the CIM and from that into the form required by the target application. Since each exchange passes from the internal schema of the source application through the conceptual schema of the integration broker to the external schema of the destination, the committee referred to this approach as a *three-schema architecture*.

Although each communication requires a two-step translation instead of the one-step translation of the two-schema architecture,

- **Number of Interfaces:** The number of interfaces to build and maintain is substantially less, growing linearly with the number of applications, rather than with the square of that number.
- **Redundancy:** The broker manages the common tasks of transforming and merging data from multiple sources, a task that would have to be done redundantly by each application in the two-schema architecture.
- **Impact Assessment:** When an application changes in a way that affects its mapping of its schema to the CIM, the only other mappings that must be examined, reverified, and possibly revised are those that contain data whose definition in the CIM has changed.
- **Scope of Knowledge:** The effect of the three-schema architecture is to hide the sources and targets of data. An application's integration architects need only know about the common information model; they need not know the sources and targets of its

contents. Hence, those sources and targets can, in most cases, be changed without critical impact on the application.

Although the three-schema architecture was published in 1977 (Jardine), it was not extensively implemented for several reasons:

- It increased the scope and complexity of the documentation to be developed and managed in order to keep track of all the schemas and mappings.
- The database technology for which it was intended did not scale well to support multiple applications with different external schemas.
- The change management practices needed to approve a change to the common information model did not meet project schedule requirements.
- Commercial products were being developed that utilized their own embedded databases that did not rely on the conceptual schema.

The end result was that database applications could be delivered faster without using a three-schema architecture, and the recommendation languished.

Today, the common information model has re-emerged as a viable architectural approach, not as an architecture for physical databases but as an approach to achieving application integration. Variations on this approach to achieving the goals of the CIM usually offer one or more of three features:

1. Ontology
2. Standard Exchange Format
3. Integration Framework

## Ontologies

An *ontology* is a specification of the information appropriate to a particular business process or domain. It defines the entities that can be the subjects of information, the properties of those entities, the relationships among them, and in some cases the fundamental operations that can be performed among them. Like the CIM, an ontology is neutral with respect to applications and technology. Moreover, although it is typically drawn from the vocabulary and practice of a particular community of users, it is also like the CIM, available for use outside that community, and may overlap with the ontologies of other communities. The CIM is essentially an integrated ontology that embraces the ontologies of all the communities that need to share information. A number of standards (described below under Standard Exchange Format) pro-

vide *languages* for building ontologies, rather than ontologies themselves.

An ontology typically leaves undefined the mechanisms by which information based on the ontology is exchanged. That is left to those who implement the ontology. However, some ontologies are embedded in an architectural framework that provides standard mechanisms for exchanging information. This is particularly the case for certain industry standards such as Aerospace Industries Association (AIA) (2004) and ISO TC184/SC4 (2004).

## Standard Exchange Format

Some industry organizations are developing standards for how to define and exchange information based on agreements about common information. They provide standard languages and mechanisms, and leave it as a separate exercise, which they often facilitate, to specify the information content to be expressed in those languages.

### United Nations Directories for Electronic Data Interchange for Administration, Commerce and Transport (UN/EDIFACT)

UN/EDIFACT established some of the original rules for global electronic data exchange (EDI). Although the technologies that implemented these UN standards are being replaced, the ontologies on which they are based constitute the bulk of most of the modern standards.

### Open Applications Group (OAG)

The Open Applications Group (OAG) provides a process for integration that depends on the exchange of XML documents that conform to the OAG standard for Business Object Documents (BODs) (OAG, 2002).

### Organization for the Advancement of Structured Information Standards (OASIS)

The Organization for the Advancement of Structured Information Standards (OASIS) defines standards for Electronic Business using eXtensible Markup Language (eXML) (OASIS, 2001) and Universal Business Language (UBL) (OASIS, 2004), both neutral languages for the definition of common information for inclusion in the CIM.

### World Wide Web Consortium (W3C)

The World Wide Web Consortium (W3C) has developed the Semantic Web (W3C, 2003), which “provides a common framework that allows data to be shared and reused

across application, enterprise, and community boundaries. It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners. It is based on the Resource Description Framework (W3C, 2004), which integrates a variety of applications using XML for syntax and URIs for naming.” The Semantic Web presumes the Internet as its only infrastructure foundation. It is based on formal languages for specifying ontologies to govern the interchange of information over the Internet:

## Integration Frameworks

The standards mentioned above try to achieve the objectives of the CIM by standardizing the content and/or format of the exchange package, and leaving the question of how a component uses or produces that package entirely to its implementation. An alternative approach attempts to optimize the integrated system by exploiting specific technologies.

### Data Management Task Force (DMTF)

The Data Management Task Force (DMTF) (2004) specifies an object-oriented framework for implementing the CIM as a collection of objects that encapsulate the routing of method invocations from client to target and back.

DMTF is the only standard described herein that actually uses the expression “Common Information Model”, which it defines as follows:

*CIM provides a common definition of management information for systems, networks, applications and services, and allows for vendor extensions. CIM's common definitions enable vendors to exchange semantically rich management information between systems throughout the network.* (DMTF, 2004)

*[An information model is an] abstraction and representation of the entities in a managed environment—their properties, operations, and relationships. It is independent of any specific repository, application, protocol, or platform.* (DMTF, 2003, p. 6)

The ontology embedded in the DMTF CIM currently focuses on components of the computing infrastructure. However, the definitions, techniques, and supporting infrastructure are applicable regardless of the system whose elements need to be managed across a network.

The DMTF architecture is based on the standards of object-oriented systems promulgated by the Object Management Group (OMG). These include:

## Common Information Model

- The Unified Modeling Language (UML) (OMG, 2003a), which is used to define the elements of the CIM.
- The Meta-Object Facility (MOF) (OMG, 2002), which provides a meta-ontology that defines the kinds of elements that can be included in an ontology.
- XML Metadata Interchange (XMI) (OMG, 2003b), which specifies how the definitions of modeling elements that conform to MOF can be exchanged between tools as XML files.

## THE COMMON INFORMATION MODEL IN INFORMATION SYSTEMS INTEGRATION

The key role of the CIM in all of the above industry activities is to reduce the coupling between components of an information system so that those components can be revised or replaced without a major redesign of other components in order to accommodate the change. The mapping of an application's external schema, that is, the face it presents to the computing environment, to the CIM rather than to other applications, effectively encapsulates it, in other words, hides its schema and operations. The CIM "knows" which applications have elements that correspond to its own elements, which of them are sources of values for those elements, and which are users. The middleware that implements the CIM "knows" what operations and interfaces are required to move an element of data in and out of an application under what conditions. The maps constitute a service contract that allows each component to evolve as required, as long as it continues to abide by that contract in its interactions with the system.

Within that general framework, there are a number of dimensions that might vary among different implementations of the CIM.

### Neutrality

The essence of the Common Information Model is neutrality with respect to the applications and technologies integrated through it, but that neutrality admits of degrees. The following levels of neutrality are derived in large part from John Zachman's Framework for an Information Systems Architecture (Zachman, 1987; Zachman & Sowa, 1992).

- The *Common Business Information Model (CBIM)* is the most completely neutral. It defines the information to be shared among business processes, for example, employees, organizations, products, and all their myriad attributes and relationships. It also defines the changes in those business objects as

they progress through various business processes. The elements of this model provide the semantics for the data that are exchanged among the computing systems, as well as for the data that those systems present to people.

- The *Common Data Model (CDM)* or *Common Object Model (COM)*, depending on the extent to which object-oriented concepts are used in specifying these common assets, is less implementation-neutral in that it specifies the *data* that represent the elements of the CIM. The CDM provides an application (and technology) neutral vocabulary and rules of representation for specifying the contents of the data that flows among computing components and people. The elements of the CDM are mapped to elements of the CIM in order to provide well-defined semantics for data. The mappings between application schemas and the CIM are ultimately to their CDM counterparts because that mapping must not only reflect semantic alignment but also representational alignment.
- The *Common Data Model Implementation (CDMI)* is a manifestation of the CIM in a given technology. Integration architects must choose whether to implement the CIM as a physical data collection, or alternatively as only a virtual collection that exists only ephemerally in transformations that are derived from the mappings of applications schemas to the CDM. Among the possible physical implementations of the CIM are the following:
  - Common objects in an integration broker, that is, middleware that transforms data into and out of the common objects.
  - Common tables in a data warehouse, that is, a database that persists data states internally for use by other applications.
  - Common objects in an application server, which may be the basis for common processing as well as for Web services, which aggregate those objects into the coarse-grained units that are suitable for business-to-business (B2B) interactions.
  - OAG Business object documents (BODs) (OAG, 2002) transmit a chunk of the CDMI through the network from one application to another.

Each of these implementations of the CDM is encapsulated from the applications that rely on them for data exchange. The applications connect only to the adapters that transform data into and out of the application. What the internal representation is between source and target adapters is irrelevant to those applications.



The use of a subset of the CDM as a schema for middleware has a number of advantages:

- It reduces the schema design effort.
- It reduces variation among the schemas of the various technologies that support integration.
- It provides a roadmap for interface design.

## Scope

The *scope* of the CIM defines the information to be comprehended, that is, its ontology, thus the kinds of information that can be exchanged through it.

No CIM can provide a *theory of everything*. Early attempts to implement shared databases presumed that development of the system would begin with an agreement on the schema for the corporate database, which would then provide concepts and guidelines for all of the components. Such a goal proved elusive. The very effort of seriously trying delayed needed applications.

The same applies to current manifestations of the Common Information Model. An *enterprise* CIM might be an ideal target, but it is unlikely to be achieved as an early milestone in the integration process. Instead, integration will proceed incrementally with a number of autonomous integration projects developing their own *local* CIMs in parallel with guidelines in place to improve the likelihood that local CIMs can be integrated into more comprehensive CIMs when the business case demands it.

## Process

The use of the Common Information Model as the basis for large-scale integration is too new a practice for the processes around it to be well-founded. Thus the suggestions below should be understood as a starting point to be refined with experience.

## Normalization

A canonical form is a standard for defining elements of the CIM. The most thoroughly developed approach to canonical form is E.F. Codd's (1970) theory of *normalized data*. Codd's theory was directed at database technology and was the mathematical foundation for relational databases. However, the mathematics of normalization applies to any collection of data designed for sharing across multiple applications.

Normalization is a technique applied to collections of data with relatively minimal documentation. What was required was only the knowledge of *functional dependencies* among data elements, that is, what values of what data elements assured the uniqueness of the values of other

data elements. Normalization partitioned data elements into subcollections whose values were determined by the same set of determining elements, or *keys*.

Normalization as a technique was reinforced by the development by Peter Chen (1976) and others of the *Entity-Relationship* approach to data model, which tended to yield data structures that converged with those produced by normalization. ER modeling can be understood as the model theory that provides the formal semantics for normalized data.

The value of normalization to the development of the CIM is that it assures that any given type of data is to be found in only one place in the collection. Indeed, normalization can be understood as a process for removing redundancies from data collections, thereby removing anomalies that complicate updates to the same data in multiple places. Normalization simplifies the mapping with application schemas by assuring non-redundancy in the CIM.

## INITIALIZING THE COMMON INFORMATION MODEL

Throughout the life of the CIM, the relationships implicit in Table 1 must be maintained:

- Common data must be mapped to the common information that provides its meaning (its semantics).
- Business process information must be mapped to the common information to provide a basis for information flow across processes and to validate definitions of common information against the rules of business processes.
- Application data must be mapped to business process information to provide meaning to it in the context of supporting the process.
- Application data must be mapped to common data to support the exchange of data through CIM-based middleware.

This cycle of relationships assures the consistency of information as the data that represent it flows among applications through the CIM. The cycle also indicates

Table 1. CIM relationships

Common Information (CIM)	Business Process Information
Common Data (CDM)	Application Data



## Common Information Model

the sources from which the CIM and the CDM are to be derived.

- **Business process vocabulary and rules:** These come from the professional disciplines on which the processes are based, as refined through the experience and practice of those disciplines within an enterprise.
- **Application data schemas:** These are typically part of the documentation of the applications and reflect the developers understanding of the disciplines and processes they are intended to support.
- **Industry standards:** Standard data models, such as those mentioned above, reflect a consensus on specific application domains and can be used as a subset of the CDM and CIM. Well-established standards are often supported by applications interfaces, which simplify the task of mapping the application to a CDM that includes them.
- **Middleware components:** Middleware products often include common data definitions that result from their vendor's efforts to integrate applications.

Although no one of the sources should be taken as final and binding on the CIM, each provides valuable input to the process of building the CIM and CDM.

## EVOLUTION OF THE COMMON INFORMATION MODEL

As applications are added, replaced, or retired, the CIM and CDM will evolve to reflect the changing collection of information that the system makes available. In most cases, when the schema of a new application is integrated into the CIM, it results in a few additional attributes and relationships being added to already defined entities. Occasionally, especially in the early phases of CIM development, whole new entities are added with new relationships. Neither of these changes normally affects existing interfaces of the CIM to the applications that had already been integrated. There are two situations in which a change of applications can be disruptive:

- *An application is retired which is the only source of data used by other applications.* This poses the dilemma of either finding a new source for that data or persuading the owners of the client applications that they no longer need it.
- In the first case, the new source can be integrated into the system, and other applications are not disrupted.

- In the second, all of the applications and interfaces, as well as all of the business processes, that require the deprecated data must be revised.
- *An application is added which requires the CIM to be restructured.* For example, a geographical analysis application is added which treats streets, cities, and states as interrelated first-class objects, rather than mere text-valued attributes. In order to apply the application to existing addresses, common objects that contain addresses will have to be revised as interrelated structures of objects. As a consequence, the maps from old applications will have to be revised to accommodate the new structure. Since the map is part of the CIM service contract, this can impose a significant procedural challenge. A mitigating approach, which this author has not yet been able to test, is to leave the old CIM structures in the CIM with appropriate mappings to the new structures, as *sunsetting* elements or as *views*, for use in existing interfaces until such time as those interfaces have to be revised for other reasons.

## THE FUTURE OF THE COMMON INFORMATION MODEL

The use of the CIM as a basis for integrating applications is a new practice for most application architects. It requires an investment in tools, techniques, and configuration management that has not been required by traditional application development. Given the pressures on costs and schedules that drive applications, many projects will resist such significant changes to their processes. On the other hand, there are a number of other forces that may overcome this inertia:

### Business Integration

Business consolidation, restructuring, partnerships, and Web-based interactions are changing the face of enterprises. Competition is forcing companies to outsource processes and to engage in digital business-to-business (B2B) interactions in order to reduce costs. Companies that cannot engage in such interactions risk loss of business.

Typically, standard B2B interactions, whether they be implemented by standard messages or by Web services or something else, cannot be fully managed by a single application and require interoperation among several applications. The use of a CIM to manage the unpacking, workflow, and assembly of these standard B2B interac-

tions is expected to simplify and reduce the cost of that interaction.

## Technology

Middleware vendors are competing aggressively to support businesses in their integration activities. Some middleware products already offer significant support for a CIM-based architecture, and others are beginning to use the rhetoric.

## Service-Oriented Architecture

Platform vendors are competing aggressively to support what is being referred to as a service-oriented architecture, which relies on communication through Web services. Currently, the marketing rhetoric suggests that individual applications will expose their data and operations through their own Web services. The implication is a two-schema architecture in which applications will directly invoke the services of another application. This makes every object exposed by a Web service a separate interface to which other applications can bind. Moreover, applications are likely to expose very similar objects with significant semantic differences that cannot be determined from information in the directories that define the services. It remains to be seen whether enterprises can develop semantically consistent systems of such applications without facing the maintenance penalties described above for point-to-point interfaces.

An alternative is to forego application-specific Web services and to develop a service-oriented architecture around a CIM. This allows the enterprise to specify precisely in a single place the semantics of the objects it is exposing to the world through its Web services. It also avoids redesigning a large number of existing applications to achieve a service-oriented architecture.

## Model-Driven Architecture

The Object Management Group (OMG) is actively promoting what it calls a Model Driven Architecture (MDA).

*MDA<sup>®</sup> provides an open, vendor-neutral approach to the challenge of business and technology change. Based firmly upon OMG's established standards [Unified Modeling Language (OMG, 2003c), Meta-Object Facility (OMG, 2002), XML Metadata Interchange (OMG, 2003b), and Common Warehouse Metamodel (OMG, 2001)], MDA aims to separate business or application logic from underlying platform technology. Platform-independent applications built using MDA and associated standards can be realized on a range of open*

*and proprietary platforms, including CORBA<sup>®</sup>, J2EE, .NET, and Web Services or other Web-based platforms. Fully-specified platform-independent models (including behavior) can enable intellectual property to move away from technology-specific code, helping to insulate business applications from technology evolution, and further enable interoperability. In addition, business applications, freed from technology specifics, will be more able to evolve at the difference pace of business evolution. (OMG, 2003a, MDA)*

The ability of MDA tools to generate working code from models and design and implementation profiles has been demonstrated. Whether they can scale to produce fully integrated systems of applications remains to be seen. One of the challenges lies in the complexity of the models that will be required at multiple levels to specify these systems and analyze their mutual impacts. Although the CIM does add a complex component to the mix, the resulting complexity of the set of models that specify the whole system is likely to be less, since it results in an architecture not burdened by the N-Squared Problem.

## CONCLUSION

The Common Information Model is a technique for insulating the details of the schemas of interoperating applications one from the other, allowing each to evolve while continuing to communicate. It is an integrated approach to reducing the overall complexity of systems, making them more manageable and adaptable to changing business requirements. The CIM manifests itself in several different industry standards with different technologies and varying ontologies. This paper has attempted to show the relationships among these standards and technologies, despite the differences in their respective vocabularies. Whether successful practices evolve that make the CIM and its supporting technology with other similar technologies remains to be seen.

## REFERENCES

- Aerospace Industries Association (AIA). (2004). Retrieved February 2, 2005, from [http://www.aia-aerospace.org/supplier\\_res/work\\_groups/ecwg/about\\_ecwg.cfm](http://www.aia-aerospace.org/supplier_res/work_groups/ecwg/about_ecwg.cfm)
- Chen, P. (1976, March). The entity relationship model: Towards a unified view of data. *ACM Transactions on Database Systems*, 1(1), 9-36.
- Codd, E.F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6),

## Common Information Model

- 377-387. Abstract retrieved February 2, 2005, from <http://www.informatik.uni-trier.de/~ley/db/journals/cacm/Codd70.html>
- DMTF (Data Management Task Force). (2003). *CIM concepts white paper*. Retrieved February 2, 2005, from <http://www.dmtf.org/standards/documents/CIM/DSP0110.pdf>
- DMTF (Data Management Task Force). (2004). Common information model (CIM) standards. Retrieved February 2, 2005, from <http://www.dmtf.org/standards/cim>
- Gruber, T. (n.d.). Ontology. Retrieved February 2, 2005, from <http://www-ksl.stanford.edu/kst/what-is-an-ontology.html>
- ISO TC184/SC4 (2004). Industrial Data. Retrieved February 2, 2005, from <http://www.tc184-sc4.org/>
- Jardine, D. (1977). ANSI/SPARC database model (three-schema architecture). *Proceedings of the 1976 IBM SHARE Working Conference on Data Base Management Systems*, Montréal, Québec.
- Merriam-Webster Online. (2004). Retrieved February 2, 2005, from <http://www.m-w.com/dictionary.htm>
- OAG (Open Applications Group). (2002). OAGIS: A “canonical” business language. Retrieved February 2, 2005, from <http://www.openapplications.org/downloads/whitepapers/whitepaperdocs/whitepaper.htm>
- OASIS (Organization for the Advancement of Structured Information Standards). (2001). ebXML technical architecture specification v1.0.4. Retrieved February 2, 2005, from [http://www.ebxml.org/specs/index.htm#technical\\_specifications](http://www.ebxml.org/specs/index.htm#technical_specifications)
- OASIS (Organization for the Advancement of Structured Information Standards). (2004). OASIS universal business language 1.0. Retrieved February 2, 2005, from [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=ubl](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl)
- OMG (Object Management Group). (2001). Common warehouse metamodel (CWM) specification. Retrieved February 2, 2005, from <http://www.omg.org/cgi-bin/apps/doc?ad/01-02-01.pdf>
- OMG (Object Management Group). (2002). Meta-object facility (MOF) specification. Retrieved February 2, 2005, from <http://www.omg.org/cgi-bin/apps/doc?formal/03-05-02.pdf>
- OMG (Object Management Group). (2003a). Model-driven architecture (MDA). Retrieved February 2, 2005, from <http://www.omg.org/mda/>
- OMG (Object Management Group). (2003b). XML metadata interchange (XMI) specification. Retrieved February 2, 2005, from <http://www.omg.org/cgi-bin/apps/doc?formal/03-05-02.pdf>
- OMG (Object Management Group). (2003c). Unified modeling language (UML). Retrieved February 2, 2005, from <http://www.omg.org/cgi-bin/apps/doc?formal/03-03-01.pdf>
- United Nations Directories for Electronic Data Interchange for Administration, Commerce and Transport. (2004). Retrieved February 2, 2005, from <http://www.unece.org/trade/untdid/welcome.htm>
- W3C (World Wide Web Consortium). (2003). Semantic Web. Retrieved February 2, 2005, from <http://www.semanticweb.org/>
- W3C (World Wide Web Consortium). (2004). RDF/XML syntax specification (revised). Retrieved February 2, 2005, from <http://www.w3.org/TR/rdf-syntax-grammar/>
- Zachman, J. (1987). A framework for information systems architecture. *IBM Systems Journal*, 26(3), 276. Retrieved February 2, 2005, from Zachman Institute for Framework Advancement (2004) [http://zifa.dynedge.com/cgi-bin/download.cgi?file\\_name=ZachmanSystemsJournal1.pdf&file\\_path=/home/zifa.com/friends/Zachman\\_SystemsJournal1.pdf&size=15856203](http://zifa.dynedge.com/cgi-bin/download.cgi?file_name=ZachmanSystemsJournal1.pdf&file_path=/home/zifa.com/friends/Zachman_SystemsJournal1.pdf&size=15856203)
- Zachman, J., & Sowa, J. (1992). Extending and formalizing the framework for information systems architecture. *IBM Systems Journal*, 31(3), 590. Retrieved February 2, 2005, from Zachman Institute for Framework Advancement (2004) [http://zifa.dynedge.com/cgi-bin/download.cgi?file\\_name=ZachmanSystemsJournal2.pdf&file\\_path=/home/zifa.com/friends/ZachmanSystemsJournal2.pdf&size=2405114](http://zifa.dynedge.com/cgi-bin/download.cgi?file_name=ZachmanSystemsJournal2.pdf&file_path=/home/zifa.com/friends/ZachmanSystemsJournal2.pdf&size=2405114)
- Zachman Institute for Framework Advancement. (2004). Retrieved February 2, 2005, from <http://www.zifa.com/>

## KEY TERMS

**Architecture:** The organization of components of a system that enables their interacting with one another to achieve the objectives of the system.

**Common Data Model (CDM):** A definition of the data to be shared across the scope of an integrated computing system in terms that are neutral with respect to the applications and technologies that make up that system. A CDM provides a vocabulary for talking meaningfully about the data that represent the information

defined in the flow, transformation, and contents of data packages that are delivered from one application or technology to another.

**Common Information Model (CIM):** A definition of the information to be shared across the scope of an integrated system. A CIM may span multiple domains, as long as the elements of each of the domains can be mapped uniquely to an element of the CIM.

**Conceptual Schema:** A definition of the information to be shared across the scope of an integrated system. This term was used by the ANSI-SPARC Committee (Jardine, 1977) for what is referred to in this paper as a Common Information Model.

**Domain:** A scope of information definition. A domain defines a collection of information generally recognized as appropriate to a field of study, a business process or function, or mission.

**External Schema:** Used by the ANSI-SPARC Committee (Jardine, 1977) for a definition of the information required by a particular application or other component.

**Integration:** A system is *integrated* when it is sufficiently interconnected that a change to any element of the system by any component of the system is reflected appropriately, that is, according to the business rules of the system, in every other component of the system. For example, if a Human Resources system is integrated, and an employee changes his or her address through any human interface that allows it, then the new address will be shown automatically every place else in the system.

**Integration Broker (IB):** A middleware product that uses an internal (or virtual) representation of a Common Data Model (CDM) to mediate the exchange of data and data-related services among applications. An IB manages the physical, syntactic, and semantic translation of data from any application, the validation of rules of authorization and business processing, and the transport to each application and component required to preserve systemwide consistency of the virtual collection of data.

**Internal Schema:** Used by the ANSI-SPARC Committee (Jardine, 1977) for a definition of the information available from a particular database, application, or other source.

**Meta-Data:** Literally, data about data. The term is used, often ambiguously in either of two ways:

1. Data that defines the *types* of data, applications, information, business processes, and anything else of importance to those responsible for specifying, building, maintaining, deploying, operating, or using components of an information system. This kind of meta-data is typified by the contents of requirements or design specifications, or of the models produced by CASE tools, and comes in many different forms. For example, the most readily accessible meta-data for our cast of technologies might be:
  - COBOL typically provides data definitions in COPYLIBS.
  - Relational databases typically provide SQL definitions of tables.
  - XML systems provide DTDs or XML Schemas.
  - Object-oriented systems provide some form of object directory.
  - Formal models of other kinds, such as UML, might be provided, but maintained models are rare.
2. Data that describes the provenance and particulars of a specific instance of a data package. For example, the header of an HTML file might specify the title, owner, dates of creation and last change, creating application, and other properties specific to that particular instance of the HTML file.

**Ontology:** A branch of metaphysics concerned with the nature and relations of being. A particular theory about the nature of being or the kinds of existents. (Merriam-Webster, 2004). “A *specification of a conceptualization*. That is, an ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents” (Gruber).

**Schema:** A definition of the data available for use by an application or system of applications.

**Three-Schema Architecture:** Used by the ANSI/SPARC Committee (Jardine, 1977) for a data-sharing architecture in which a client application interfaces its external schema only to a conceptual schema, which is itself interfaced to the internal schemas of available sources.

**Two-Schema Architecture:** Used by the ANSI/SPARC Committee (Jardine, 1977) for a data-sharing architecture in which a client application interfaces its external schema directly to the internal schema of a source.



# Component-Based Generalized Database Index Model

**Ashraf Gaffar**

*Concordia University, Canada*

## INTRODUCTION

The performance of a database is greatly affected by the performance of its indexes. An industrial quality database typically has several indexes associated with it. Therefore, the design of a good quality index is essential to the success of any nontrivial database. Parallel to their significance, indexed data structures are inherently complex applications that require a lot of effort and consume a considerable amount of resources. Index frameworks rely on code reuse to reasonably reduce the costs associated with them (Lynch & Stonebraker, 1988; Stonebraker, 1986). Generalized database systems have further addressed this challenge by offering databases with indexes that can be adjusted to different data/key types, different queries, or both. The generalized search tree (GiST; Hellerstein, Naughton, & Pfeffer, 1995; Hellerstein, Papadimitriou, & Koutsoupias, 1997) is a good example of a database system with a generalized index, or generalized index database for simplicity. Additional improvements extended the concept of generalized index databases to work on different domains by having generalized access methods (search criteria). For example, based on the work of Hellerstein et al. (1995), Aoki (1998) provides a generalized framework that allows users to adjust the index to different search criteria like equality, similarity, or nearest neighbor search. This makes the database system customizable to not only finding the exact records, but also to finding records that are “similar” or “close” to a given record. Users can customize their own criteria of “similarity” and let the index apply it to the database and return all “similar” results. This is particularly important for more challenging domains like multimedia applications, where there is always the need to find a “close image,” a “similar sound,” or “matching fingerprints.”

The common drawback of all these improvements is having a monolithic code that allows users to only adjust the few lines of code that are meant to be customized. Outside these areas, code is often bulky and difficult to maintain or upgrade. However, with the increasing dependence on software engineering techniques, these problems can be ratified. Component-based frameworks provide solid ground for future maintenance, replace-

ment, and additions to the code in an orderly fashion that does not reduce the quality of the system (the aging effect). Customization does not have to be limited to few lines of the code. In our work, we provide a new model to redesign the generalization of database indexes using components. Unlike previous works, the customization is not done at the code level, but rather at higher conceptual levels from the early design stages. Our design is based on total decoupling of the design modules and connecting them through well-defined interfaces to build the database system. All this is done at the design level. After the design is completed, implementation is done by obtaining the already-existing commercial **off the-shelf** (COTS) components and connecting them together according to the design model. Following this design-level customization paradigm, the system can be customized as usual by the user at prespecified locations (predefined few lines of code), but more importantly, a large-scale customization is also possible at the design level. The system designer can redesign the generalized model by adding, removing, or replacing a few components in the design model to instantiate the model into a new concrete design. Afterwards, only the affected components of the source code need to be replaced by new ones. In our system, COTS components are extensively used, which dramatically reduces the cost of development. We needed to implement a few components with the same predefined interfaces where COTS components were not suitable, for example, when special concurrency control management or specific storage needs were necessary. This adds more flexibility to the model.

## BACKGROUND

Component-based software systems have received a lot of attention recently in several application domains. Reusing existing components serves the important goals of simultaneously reducing development time and cost (time-to-market) and producing quality code. Moreover, components can be added, removed, or replaced during system maintenance or reengineering phases, which leads to faster and better results (Sanlaville,



Faver, & Ledra, 2001). Quartel, Sinderen, and Ferreira Pires (1999) offer some useful techniques to model and analyze software components and their composition. To promote successful reuse, components have to be context-independent (Schmidt, 1997). C++ has adopted a component-based library as its standard library (STL, the Standard Template Library), which provides programming-level components that can be used at the design stage and later added as code fragments to generate a considerable part of the application code (Musser, Derge, & Saini, 2001). STL adopts a new paradigm that separates software applications into a few component types for storage and processing. Using the STL paradigm at the early phases of conceptual design of the generalized database indexes facilitates the design and implementation of the whole systems by iteratively adding arbitrary combinations of STL types.

## STL BUILDING BLOCKS

We built a modular framework for a generalized database by applying the STL (ANSI C++) modularity concept (Austern, 1999) in the analysis, architecture, design, and interface stages. We developed a new set of models for an index in different application domains; linearly ordered domain, general domain (with both depth-first and breadth-first access methods), and eventually similarity search domain.

Using coherent, decoupled building blocks allows us to locate and replace a few blocks in each model to obtain a new model with no major impact on the rest of the system design. This makes modifications limited to specific regions of the system. A wealth of STL modules can be used in the replacement process. The framework also allows for introducing new compatible modules as needed.

STL is based on separating *algorithms* from *data structures* as two different types of generic building blocks (Breymann, 2000). It also introduces other generic building blocks to complete all the abstract design blocks (e.g. iterators, functors, container adaptors). They work as “adaptors” or “glue” to allow for building a large system using arbitrary combinations of these blocks. This emphasizes code reuse, a fundamental concept in modern software engineering. STL allows for new building blocks to be written and seamlessly integrated with the existing ones, thus emphasizing flexibility and extendibility. This makes STL particularly adaptive to different programming contexts, including algorithms, data structures, and data types. We briefly explain some of the major building blocks of STL that we used in the design and implementation of the index system.

## Containers

A container is a common data structure that stores a group of similar objects, each of which can be a primitive data type, a class object, or—as in the database domain—a data record. The container manages its own objects: their storage (see the Allocators section) and access (see the Iterators section). The stored objects belong to the container and are accessed through its interface. Each container provides a set of public data members and member functions to provide information and facilitate the access to its elements. Different container types have different ways of managing their objects. In other words they offer different sets of functionality to deal with their objects.

## Iterators

Containers do not allow direct access to their stored objects, but rather through another class of objects called iterators. An iterator is an object that can reference an element in a container. It allows programmers to access all elements in a sequential, random, or indexed way depending on the type of the container.

## Algorithms

Iterators separate the processing *logic* from the *container types* and *element types* stored inside them. The same algorithm, written once, can be applied to different containers with different stored elements by having the algorithm code deal with iterators that can access the containers. This allows for a complete implementation of generic algorithms, like search and sort, without knowing the exact type of the container (its functionality) or the type of elements stored in it.

## Allocators

As we know, containers are responsible for managing both the *access* and the *storage* of their elements. At the front end, the access is allowed by providing “suitable” iterators and a set of functions to provide information about the elements, to return a reference to an element, or to add/delete an element. At the back end, the physical storage remains an essential part to the container, allocating memory to new elements and freeing memory from deleted elements. Normally the user need not worry about storage management when using a container. This is done “behind the scenes” without any user intervention. This greatly simplifies the use of containers. Frequently, however, some applications require a

special kind of storage management and cannot simply work with the standard memory allocation. If the container was made directly responsible for the back-end storage, this would have meant that those special applications would need to discard the STL containers altogether and write their own special containers, defeating the purpose of code reuse, efficiency, and elegance of STL building blocks in this area of design. Therefore, to be able to serve those special cases as well, while still keeping the containers simple for everyone else, STL made a further separation of its building blocks. The container is not directly responsible for the storage. An *allocator* class was made responsible for the physical storage of the container elements, and each container handles storage by simply asking an allocator object to do just that. To keep things “behind the scenes” for users, each container uses a “default” general-purpose allocator for the storage needs, unless given a specialized allocator to handle the job.

**BUILDING BLOCKS’ CONNECTIVITY**

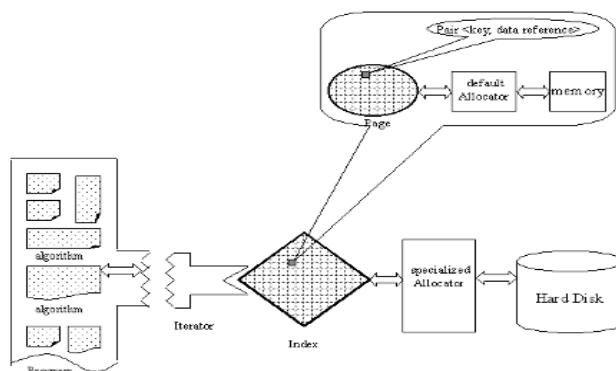
STL provides us with a complete set of compatible building blocks that can be connected together—using logical connectivity rules—to build complex systems. Using these blocks, we can produce a “blueprint” design of the system components, which can be drawn as a directed graph. The nodes would be STL blocks and the edges would be the allowed connections between them. Semantically, a connection between two nodes would represent a “use” relationship between the two STL blocks represented by the nodes. Connecting these blocks is not, however, done at random. We cannot simply connect any two blocks of our choice and put them to work. There are certain rules that apply when trying to connect two blocks together.

**THE MODULAR INDEX SYSTEM**

Figure 1 shows the building blocks of the indexed database model. In this system, integrated in a larger application (Butler et al., 2002), the index is seen as a container that provides an iterator to its database contents. The only way to interact with this container is through its iterator, which provides controlled access to the elements of the container.

Due to the huge number of entries in a typical database, the entries are paged rather than manipulated individually (Folk & Zeollick, 1992). The *index container* is therefore an index of pages, recursively built as containers with their own iterators, allocator, and memory man-

Figure 1. Subsystem view of the index framework



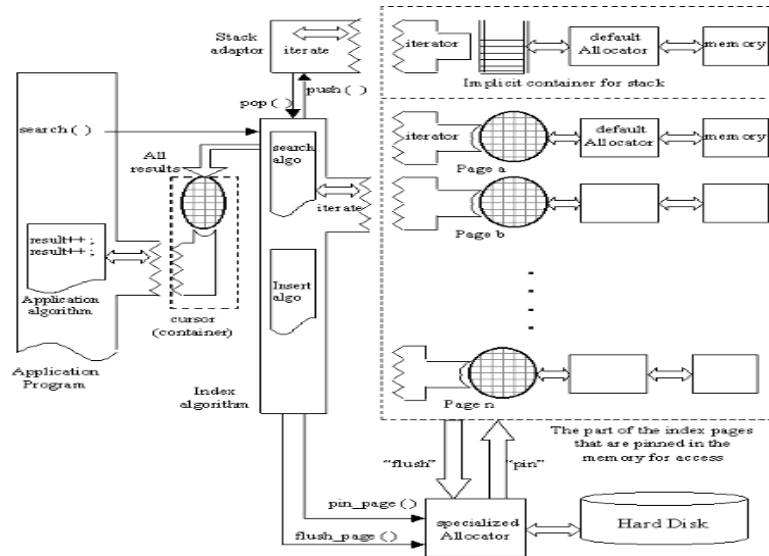
agement schema. In other words, the elements that the index stores are of type “page.” Each *page* is itself a container on a smaller scale. While the database is a container of pages that typically resides on hard disk, the page is a memory-resident container of *pairs of key and data references*. Again, the page is made independent of the types of both *key* and *data reference* by having them as two templates passed to the page in the implementation phase. Therefore the same page can work with any key type and data reference type that are passed to it.

The page has a *page allocator*, responsible for memory management like storage allocation and deallocation. The page uses a standard in-memory allocator by default. This allocator assumes that the page is entirely stored in memory, which is a reasonable assumption regarding the concept of a page in a database. Therefore it is likely for many systems to use the default page allocator. If the system has special needs for page storage, like specific space or time management, the designer can provide a special page allocator without affecting the design of the other parts of the system. The page container is unaffected by the type of allocator used, as long as it satisfies the standard interface to provide the page with all the services it needs.

Pages can iterate through their contents, searching for specific pairs of data that satisfy some key according to a similarity criteria provided by a search function. Again, the decoupling of components allows us to write any similarity criteria in the form of an algorithm module that decides if data entities are similar. Once a location is found, the corresponding pair is returned to the object asking for this service. This implies that the iteration process is done internally, with no page iterator needed to be visible for the outside world. The



Figure 2. Overall view of the system



page, however, still provides an iterator for other objects to allow for expansion.

At the code level, pages are passed to the index as “template” type during the first time the database is populated with real data or pointers to data locations. This makes the index independent of the page design and allows the index to work with any page type that is passed to it. *The index iterator* is therefore iterating through pages of any type, one page at a time. This is exactly the concept of database indexes: *paged indexes* to facilitate access and improve performance.

Unlike the in-memory page allocator, the index uses an *index allocator* to manage the storage of its own elements, the index pages. In practical application the index exists permanently on nonvolatile storage, like a hard disk or other mass storage media, since it is normally too large to fit entirely in memory. This means that the container will use a *specialized allocator* that takes the responsibility of retrieving the page from the storage into memory for access and controlling the different objects accessing the same page concurrently. This will ensure data integrity by applying a suitable access and locking policy (pinning the page). After the pages have been modified, they also need to be updated in the physical storage (flushing the page) by the allocator. A default, in-memory allocator is provided for simple applications where the whole index can fit in memory at one time. It is up to the system designer to use it or override it by providing a storage-dependent specialized allocator. Figure 2 provides the overall system model.

The separation between the container and the allocator allows for the overall system to be completely independent of the physical storage of the data. The container uses the standard allocator interface to ask for storage services without any knowledge of the real storage dynamics. A stack is added to the system to support general domain applications where each internal page can point to several nonlinearly ordered pages. In this case, all matching entries need to be temporarily pushed into a stack. Again, the stack is built as another container with its iterator and allocator. A stack is a specialized type of container that is obtained in the STL model by providing a stack adaptor that limits the access of a more flexible container to the access operations allowed by a stack, namely, `push()` and `pop()`. This is a good example of the use of adaptors provided by STL.

Finally, the query results are pushed into a cursor, which is yet another container assembled from STL components. In the end, we were able to provide a fully functional system with the careful design of models using STL components.

## THE MODEL LIFE CYCLE

As in the case of building generalized database systems, the system model will evolve through four major phases during its life cycle:

1. **The Generic Design phase**, where the generic model framework is designed once (in this case, our system offers the generic design layout).

2. **The Adaptation phase**, where *system architects* identify the requirements of a specific domain application and determine the corresponding system components needed in the model. *System developers* then instantiate this particular system, in which all generic domain attributes have been made specific, like the data and key types, the type of database tree structure, the access methods, the need for stacks, etc. We have adapted the generic model to implement several concrete designs supporting different data, trees, and access methods.
3. **The Data Loading phase**, where the initial loading of data into the system will determine the structure of the database and, hence, the access methods supported by the index. Data loading is generally carried out by the *database administrator* and/or *system developer*. It is done in one of two ways. For a small amount of data and for testing and debugging purposes of the system, the normal interface of the index, namely, the *insert* method, is used to insert the necessary data into the database. This is referred to as *transactional data loading*. Despite the fact that it needs to be done only once while populating the database, it can still be expensive when fully loading a large-size database. For this latter case, data can be loaded from an ASCII file into the database by writing a small program (often called a loader) that reads the whole ASCII file, separates the data records, and loads them directly to database pages. This approach is referred to as *bulk loading* and is capable of efficiently loading a large amount of data into the database index.
4. **The Use phase**, where the database is accessed by other applications through its standard interface. *Application programmers* are interested only in this phase. The interface supports common database transactions like insertion, deletion, and updating.

Tree-based database indexes can contain the actual data in their nodes and leaves (B-trees) or only in their leaves (B+-trees and the like). Indexes can also be totally separated from physical data (like the case of multimedia applications, where each record can be a large image, a sound file, or even a whole video session). In this case, actual data is typically handled through pointers, which are much smaller in size than the actual records. The system architect determines if the index will load the actual data or will only load pointers to the actual data (along with the appropriate keys). In our modeling concepts, both cases are supported, as they are seen as different data types loaded to the index.

## FUTURE TRENDS

Database systems are fundamental to any industrial quality database. So far, most commercial databases have used their proprietary design and implementation. Generalized index frameworks help in producing industrial quality databases; however, they suffer from the complexity of their code and the lack of modularity in their design. As component-based development is proving to be powerful and reliable, the emergence of software applications and frameworks that rely on components is gaining momentum. We already have CORBA, JavaBeans, and Web services that are following this trend. So far it has been mostly adopted in small-scale database implementations, but the trend to implement them on full-scale industrial database systems is evolving. Nystorm (2003) provides a good example of a component-based real-time application that demonstrates these concepts in database domain.

## CONCLUSION

We have applied the component-based system concepts to introduce the paradigm of modular design to the database system in order to improve the quality of design. We used the STL components to build a generalized model. The STL concept allowed us to put a model for the system from the early design phases. We put components together to build subsystems and to further connect the subsystems together iteratively until we had a full index model. Implementation is manageable by collecting the components together according to the architectural design model and instantiating them to the final code. The system built after this architectural model is easier to implement than a non-component index of same capability; however, it still requires a significant level of programming. The system can be adapted to different contexts and requirements by re-writing only some of the components. All the basic components used in the system (containers, iterators, and algorithm components) are already provided as a complete COTS implementation, which greatly reduced the effort needed for the development of the index system.

## REFERENCES

Aoki, P. M., (1998). Generalizing search in generalized search trees. In *Proceedings of the 14th IEEE International Conference on Data Engineering*, Orlando, FL (pp. 380-389). IEEE Computer Science Publishers.



Austern, M. H. (1999). *Generic programming and the STL*. Boston: Addison-Wesley.

Breyman, U. (2000). *Designing components with the C++ STL: A new approach to programming*. Boston: Addison-Wesley.

Butler, G., Chen, L., Chen, X., Gaffar, A., Li, J., & Xu, L. (2002). The know-it-all project: A case study in framework development and evolution. In K. Itoh & S. Kumagai (Eds.), *Domain oriented system development: Perspectives and practices* (pp. 101-117). London: Taylor & Francis Publishers.

Folk, M. J., & Zeollick, B. (1992). *File structures* (2nd ed.). Boston: Addison-Wesley.

Hellerstein, J. M., Naughton, J. F., & Pfeffer, A. (1995). Generalized search trees for database systems. In *Proceedings of the 21st International Conference on Very Large Databases* (pp. 562-573).

Hellerstein, J. M., Papadimitriou, C. H., & Koutsoupias, E. (1997). Towards an analysis of indexing schemes. In *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (pp. 562-573).

Lynch, C. A., & Stonebraker, M. (1988). Extended user-defined indexing with application to textual databases. In *Proceedings of the Fourth International Conference on Very Large Databases* (pp. 306-317). San Francisco: Morgan Kaufmann.

Musser, R. D., Derge, J. G., & Saini, A. (2001). *STL tutorial and reference guide: C++ programming with the Standard Template Library*. Boston: Addison-Wesley.

Nystorm, D. (2003). *COMET: A component-based real-time database for vehicle control-system*. Vasteras, Sweden: Malardalen University, Computer Science and Engineering Department. Malardalen University, Vasteras, Sweden.

Quartel, D. A. C., Sinderen, M. J. van, & Ferreira Pires, L. (1999). A model-based approach to service creation. In *Proceedings of the seventh IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems* (pp. 102-110). Cape Town, South Africa: IEEE Computer Society.

Sanlaville, R., Faver, J. M., & Ledra, Y. (2001). Helping various stakeholders to understand a very large software product. In *European Conference on Component-Based Software Engineering, ECBSE*, Warsaw, Poland.

Schmidt, R. (1997). Component-based systems, composite applications and workflow-management. In *Proceedings of Workshop Foundations of Component-Based Systems*, Zurich, Switzerland (pp. 206-214).

Stonebraker, M. R., (1986). Inclusion of new types in relational database systems. *Proceedings of the second IEEE International Conference on Data Engineering*, Washington, DC (pp. 590-597).

## KEY TERMS

**Access Methods:** In the database domain, indexes are designed to access data that are stored in a specific structure. The type of data and the type of the structure used determine the procedures followed by the index to access these data, which is referred to as the access method.

**Aging Effect:** Software artifacts evolve over time due to the changes in domain requirements, platform, and even language. After their release, software needs to be modified regularly. These modifications introduce errors to the software and reduce its overall quality over time, which is often called the aging effect.

**ANSI C++:** Complete set of standards provided by the American National Standards Institute in collaboration with ISO standardization committee to define an industrial standard for the C++ programming language.

**COTS:** Commercial off-the-shelf software components that are generic enough to be obtained and used in different applications. They are often well designed and well implemented to offer good performance.

**Logical Connectivity Rules:** In STL, certain logical rules are defined to allow for connecting components to produce semantically correct artifacts. Incorrect connections are not allowed.

**System Reengineering:** When an existing software system requires major changes, typically by altering its original design, specific methodologies are used to help modify the design successfully. This process is referred to as system reengineering.

**Template:** At the programming level, a template allows programmers to fully write and debug their code without specifying certain types of variables. Later the template can be instantiated by providing it with the necessary variable types as needed. The same template can be instantiated several times with different sets of variable types.



# Consistency in Spatial Databases

M. Andrea Rodríguez-Tastets

University of Concepción and University of Chile, Chile

## INTRODUCTION

During the past several years, traditional databases have been enhanced to include spatially referenced data. Spatial database management (SDBM) systems aim at providing models for the efficient manipulation of data related to space. Such type of manipulation is useful for any type of applications based on large spatial data sets, such as computer-aided design (CAD), very large scale integration (VLSI), robotics, navigation systems, and image processing.

Spatial data applications, (in particular, geospatial applications) differ from traditional data applications for the following reasons (Voisard & David, 2002):

- spatial information deals with spatial and nonspatial data, which implies that the definition of spatial data types be closed under the operations applicable to them;
- many spatial data are inherently uncertain or vague, which may lead to conflicting data (e.g., the exact boundary of a lake or pollution area is often unclear);
- topological and other spatial relations are very important and are usually implicitly represented;
- data are highly structured by the notion of object aggregation;
- user-defined operations require an extensible underlying model; and
- functions exist at both a low level of abstraction (e.g., points, lines, polylines) and a high level of abstraction (e.g., maps, thematic layers, configurations).

Spatial databases often deal with different kinds of data imperfections, which can be classified into uncertainty, imprecision/vagueness, incompleteness, and inconsistency (Pason, 1996). Whereas uncertainty, imprecision, vagueness, and incompleteness are usually seen as different types of data inaccuracy that arise from problems in data collection, inconsistency is the kind of data imperfection that results from the existence of contradictory data.

Contradictory data in spatial databases arise from different forms of errors (Cockcroft, 1997). A primary source of errors that can give rise to contradictions is

the inconsistency generated by conflicting descriptions of locations or the characteristics and qualities of spatial features. This kind of inconsistency is commonly associated with problems of positional or data inaccuracy. A secondary source of errors that can result in contradictions is the mismatch between stored data and structural or semantic consistency rules underlying the model of reality (e.g., a region that is represented by a polyline that is not closed). Database designers usually attempt to avoid this second kind of error by enforcing integrity constraints.

Research progress on consistency in spatial databases has been the result of an interdisciplinary effort. This effort has dealt with ontological issues (Frank, 2001) concerning the definition of semantic and topological consistency. It has also considered the appropriate conceptual frameworks for analyzing spatial consistency, the specification language of integrity constraints, and the design of computational–geometry algorithms to implement consistency checkers.

The following section describes models for defining consistency of spatial data that focus on topological consistency in the presence of multiple representation levels or in the integration of heterogeneous databases. Subsequently, the specification of integrity constraints in spatial databases is discussed to complete the background for presenting challenges and future trends in the treatment of consistency in spatial databases.

## BACKGROUND

The conceptual bases for modeling consistency are topological and other geometric characteristics of spatial objects. In particular, spatial primitives and spatial relations are fundamental in the definition of consistency rules that enforce a particular model of space. Spatial primitives depend on the classical distinction between *field-based* and *entity-based* models (Shekhar, Coyle, Goyal, Liu, & Sakar, 1997). Each of these approaches to modeling space implies spatial primitives with their own definitions and rules. For example, the field-based approach to space includes the definition of tessellations, isolines, and triangular irregular network, with their corresponding definition rules. Likewise, the

entity-based approach includes definitions rules for lines and polygons.

In addition to spatial primitives, spatial relations play an important role in spatial information systems, because such relations describe spatial information and are used as the spatial criteria for querying spatial databases. Even though positional information is often a basis for specifying the spatial component of objects, spatial relations such as adjacency and containment do not require absolute positional data. Common spatial relations are typically grouped into three kinds: *topological*, *orientation*, and *distance*. Topological relations deal mainly with the concept of connectivity and are invariant under topological transformations, such as rotation, translation, and scaling. Orientation relations presuppose the existence of a vector space and are subject to change under rotation, while they are invariant under translation and scaling. Distance relations express spatial properties that reflect the concept of a metric and, therefore, change under scaling, but are invariant under translation and rotation.

Models for each kind of spatial relation exist, which have led to a productive research area referred as qualitative reasoning (Stock, 1997). Such models give definition rules for each type of spatial relations as well as consistency in the combination of spatial relations. In particular, there is a comprehensive method for analyzing the topological consistency of spatial configurations based on the logical consistency expressed by the composition of relations (Egenhofer & Sharma, 1993). For example, given that a spatial object *A* is inside of a second object *B*, and *B* is disjoint to a third object *C*, to keep the topological consistency, *A* must also be disjoint to *C*.

Two important cases that have been addressed in the modeling of consistency that differ from basic definition rules are consistency at multiple representations and consistency for data integration. The problem of multiple representations consists of data changing their geometric and topological structures due to changes in scale. Conceptually, multiple representations may be considered as different data sets that cover the same area with different levels of detail. Within the context of assessing consistency at multiple representation levels, topological relations are considered as first-class information, which must prevail in case of conflict (Egenhofer & Clementini, 1994. Kuipers, Paredaens, & den Busshe, 1997). Multiple representation levels in spatial databases may not imply inconsistent information, but rather, merely different levels of detail or scale. In such cases, topological consistency at the level of objects and objects' interrelations must be enforced.

The modeling of topological consistency at multiple representation levels has been based on the definition of

topological invariants across multiple representations (Tryfona & Egenhofer, 1997). Examples of topological invariants are the set intersections of boundaries and interiors as well as the sequence, dimension, type, and boundedness of boundary-boundary intersections. The comparison of topological invariants is used to define the topological consistency of objects and topological relations at multiple representation levels. In addition, topological invariants and consistency-checking of topological configurations have been basis for defining consistency of composite objects at multiple representations (Egenhofer, Clementini, & Di Felice, 1994).

Spatial data sets to be integrated are assumed to contain the same features or objects that can be extracted from several sources at different times. These data may vary in reliability, accuracy and scale of representation. Thus, integrating spatial information may create conflicts due to the different representations for the same features concerning, for example, shape, dimension, and positional accuracy. In this context, different types of consistency can be distinguished (Abdelmoty & Jones, 1997): *total consistency*, which occurs when two data sets are identical; *partial consistency*, which occurs when certain subsets of two data sets are identical; *conditional consistency*, which occurs when by applying a set of functions over a data set it becomes totally consistent with respect to another data set; and *inconsistency level*, which occurs when there is nothing in common between data sets.

The common approach to integrating different representations has assumed that, when no further information exists about the origin of data, both representations are adopted. The idea is to merge both representations such that the resulting representation is modeled as a vague or unclear one. In modeling these unclear boundaries, three alternatives are found: *fuzzy models* (Schneider, 2000, Userly, 1996), which are based on the theory of fuzzy sets and have been applied to spatial uncertainty; *probabilistic models* (Burrough, 1996; Finn, 1993), which are based on probability theory to model positional and measurement uncertainty; and *exact models* (Clementini & Di Felice, 1996, 1997; Erwing & Schneider, 1997), which map data models for spatial objects with sharp boundaries onto spatial objects with broad boundaries.

## INTEGRITY CONSTRAINTS IN SPATIAL DATABASES

Integrity constraints enforce consistency in spatial databases. Integrity constraints must be taken into account when updating a database such that the semantics and quality of data are preserved. In the spatial domain,

integrity constraints have been mainly used for preventing inconsistency concerning constrictions with respect to rules of geometric abstractions (e.g., polygons, lines, networks) that are used in the representation of spatial objects, whereas conflicting information about positional information has been treated as a problem of data accuracy.

In addition to traditional integrity constraints concerning static, transition, and transactional aspects of database systems, general rules about spatial data must ensure consistent updating of spatial databases. A typical classification of these spatial constraints follows (Cockcroft, 1997):

- *Topological constraints* address geometrical properties and spatial relations. They may be associated with structural considerations, such as that polygons must be closed polylines, or topological conditions, such as centerlines, must meet at intersections. Considering a subset of topological constraints, Servigne, Ubada, Puricelli, and Laurini (2000) defined *topo-semantic* constraints as those that relate geometry with semantic conditions, as in the constraint that a city's administrative region must be contained within its city limits.
- *Semantic integrity constraints* are concerned with the meaning of geographic features (e.g., roads should not run through bodies of water).
- *User-defined integrity constraints* are equivalent to business rules in nonspatial database management systems—DBMS (e.g., one must have the appropriate legal permission in order to install a gas station).

Like traditional database systems, in spatial databases, constraints at a conceptual and logical level are inherited by the implementation or physical level. These constraints are translated into a proprietary scripting language or into explicit constraints coded into the application programs. At a logical level, definitions of constraints based on topological relations have been defined (Hadzilacos & Tryfona, 1992). These constraints use a model for defining topological relations that are independent of absolute positional data, a model called the 9-intersection model (Egenhofer & Franzosa, 1991, 1994). In this model, spatial relations, and any geometric operators, can be declared as atomic topological formulae and combined in topological sentences to specify topological constraints.

With an object-oriented perspective, an extension of the OMT (object modeling technique) model to OMT-G (object-oriented data model for geographic applications) provides primitives to represent spatial data with their respective integrity constraints (Borge, Davis, & Laender,

2001, 2002). Constraints are defined by using spatial primitives and constructs for spatial relationships. Within such an object-oriented data model, constraints are encapsulated as methods associated with spatial classes. Some of these constraints, in particular aggregation rules, are formulated with operators over parts (i.e., spatial objects) and quantifications over sets, such that they are not expressed in first-order logic.

The specification of constraints can be done by different techniques of knowledge representation. Some attempts have been made to provide end users with easy mechanisms that hide the logic involved in specifying constraints (Cockcroft, 2001; Servigne, Ubada, Puricelli, & Laurini, 2000). A proposal for constraint specification allows users to define constraints in an English-like language (Servigne, Ubada, Puricelli, & Laurini, 2000). Basic components of the language are entity classes, relations, and qualifiers (e.g., forbidden, at least  $n$  times, at most  $n$  times, or exactly  $n$  times). Following the same idea, a more recent work (Cockcroft, 2001) extends the previous specification to include attribute values in the topological constraints. For example, "a butterfly valve must not intersect a pipe if the diameter of the pipe is greater than 40 inches." Interfaces with high-level languages to specify integrity constraints are standalone software tools that are integrated with a geographic information system (GIS).

## FUTURE TRENDS

The treatment of consistency in spatial databases has focused mainly on defining integrity constraints and enforcing them for data manipulation. Although there are advances in issues concerning formal models for detecting and defining inconsistency in different processes of spatial data manipulation, such as multiple representation and data integration, most of the effort is still at the conceptual level or in isolated implementations. In this context, there exist important issues and challenges in the treatment of consistency for current spatial databases, which are summarized as follows:

- **Inaccuracy as inconsistency:** The conflicting information that arises from data inaccuracy (e.g., conflicting positional information) may be treated as inconsistency if there exists an explicit integrity constraint that reflects this kind of conflicting information. What will be treated as a problem of data inaccuracy or a problem of inconsistency is still something that needs clarification.
- **Space does not automatically generate attribute constraints:** The spatial dimension of an object

does not automatically constrain the attributes of the object or the attributes or spatial components of other objects. It should be, however, an easy way to specify spatial integrity constraints.

- **Partial consistency:** A geometric representation can be totally or partially consistent such that queries based on spatial criteria over such representation are not necessarily inconsistent.
- **Spatial relations:** Spatial relations are usually implicitly represented and may not need positional accuracy. So, what is inconsistency with respect to objects' positional information may not be inconsistency with respect to spatial relations between objects.
- **Consistency of composite objects:** Composite objects treat aggregations of objects and impose constraints with respect to their parts to enforce consistency.
- **Application-independent vs. application-dependent integrity constraints:** Application-independent integrity constraints associated with spatial primitives can be built into the system with ad-hoc implementations. Application-dependent constraints, in contrast, require facilities for the generation of code to impose the constraints.
- **Consistency with propagation of updates:** A modification in a spatial database may cause simultaneous updates in a large number of records with consequences in the consistency of data.
- **Consistency at multiple representation levels:** Spatial databases may need to treat different levels of detail in the spatial representation. Consistency at multiple representations needs to be based on research that goes beyond considering topological relations, to incorporate, for example, orientation and distance relations.
- **Consistency across heterogeneous spatial databases:** Distributed and interoperating spatial information systems are not always designed under the same conceptual model, with the result that what is consistent in one database is inconsistent in another. Further studies are needed to analyze the consistent integration of data not only at the geometric level but also at the semantic level.
- **Management of inconsistency tolerance:** In the presence of inevitable inconsistencies in a database, it is necessary to find strategies that provide consistent answers despite the fact that the database is inconsistent with respect to a set of integrity constraints. For example, a database with conflicting positional information may still provide consistent answers with respect to topological relations.
- **Context dependence of integrity constraints:** Some integrity constraints are associated with the computational application of particular spatial operators (e.g., area and intersection). Context may determine the integrity constraint that is required for a particular use.
- **Integration of solutions:** Advances in modeling consistency in spatial databases, such as models of consistency at multiple representations and data integration, should be integrated to give solutions to real problems.
- **Efficient algorithms:** The complexity of spatial data requires efficient algorithms for implementing consistency models of spatial data.

## CONCLUSION

This article has discussed models of consistency and the specification of integrity constraints that emphasize the current issues and challenges for handling consistency in spatial databases. In this discussion, relevant aspects are composite objects by spatial aggregation, topological relations, multiple representation levels and integration of spatial databases. It is argued that consistency cannot be considered as a binary condition, with data being either consistent or inconsistent, but not both. Rather, consistency is best seen as a spectrum of relative conditions that can be exploited depending on the use of data.

Since spatial databases and spatial-temporal databases have vast domains of applications, it is expected that they will become a widely used technology. In addition, with the free access of available information, more data from different sources is available. In this context, data consistency in spatial databases is a current and challenging area of research and development.

## REFERENCES

- Abdelmoty, A., & Jones, C. (1997). Toward maintaining consistency in spatial databases. *Sixth International Conference on Knowledge Management* (pp. 293-300). Las Vegas: ACM Press.
- Borges, K., Davis, C., & Laender, A. (2001). OMT-G: An object-oriented data model for geographic information applications. *GeoInformatica*, 5, 221-260.
- Borges, K., Davis, C. & Laender, A. (2002). Integrity constraints in spatial databases. In J. Doorn & K. Rivero (Eds.), *Database integrity: Challenges and solutions* (pp. 144-171). Hershey, PA: Idea Group Publishing.



- Burrough, P. (1996). Natural objects with indeterminate boundaries. In A. Frank (Ed.), *Geographic objects with indeterminate boundaries GISDATA* (pp. 30-28). London: Taylor & Francis.
- Clementini, E., & Di Felice, P. (1996). An algebraic model for spatial objects with indeterminate boundaries. In A. Frank (Ed.), *Geographic objects with indeterminate boundaries GISDATA* (pp. 155-169). London: Taylor & Francis.
- Clementini, E., & Di Felice, P. (1997). Approximate topological relations. *International Journal of Approximate Reasoning*, 16, 73-204.
- Cockcroft, S. (1997). A taxonomy of spatial integrity constraints. *GeoInformatica*, 1, 327-343.
- Cockcroft, S. (2001). Modelling spatial data integrity constraints at the metadata level. In D. Pullar (Ed.), *GeoComputation*. Retrieved February 15, 2004, from <http://www.geocomputation.org/2001/>
- Egenhofer, M. (1997). Consistency revised. *GeoInformatica*, 1(4), 323-325.
- Egenhofer, M., E., Clementini, E., & Di Felice, P. (1994). Evaluating inconsistency among multiple representations. *Spatial Data Handling*, 901-920.
- Egenhofer, M., & Franzosa, R. (1991). Point-set topological spatial relations. *International Journal of Geographic Information Systems*, 5, 161-174.
- Egenhofer, M., & Franzosa, R. (1994). On the equivalence of topological relations. *International Journal of Geographic Information Systems*, 8, 133-152.
- Egenhofer, M., & Sharma, J. (1993). Assessing the consistency of complete and incomplete topological information. *Geographical Systems*, 1, 47-68.
- Erwing, M., & Schneider, M. (1997). Vague regions. *Symposium on Advances in Spatial Databases, LNCS, 1262* (pp. 298-320). Berlin, Germany: Springer-Verlag.
- Finn, J. (1993). Use of the average mutual information index in evaluating error and consistency. *International Journal of Geographic Information Science*, 7, 349-366.
- Frank, A. (2001). Tiers of ontology and consistency constraints in geographic information systems. *International Journal of Geographic Information Science*, 15, 667-678.
- Hadzilacos, T., & Tryfona, N. (1992). A model for expressing topological constraints in geographic databases. In A. Frank, A. Campari, & U. Formentini (Eds.), *Theories and methods of spatio-temporal reasoning in geographic space COSIT '92, LNCS, 639* (pp. 252-268). Berlin, Germany: Springer-Verlag.
- Kuipers, B., Paredaens, J., & den Bussche, J. (1997). On topological equivalence of spatial databases. In F. Afrati & P. Kolaitis, (Eds.), *6<sup>th</sup> international conference on database theory, LNCS, 1186* (pp. 432-446). Berlin, Germany: Springer-Verlag.
- Pason, S. (1996). Current approaches to handling imperfection information in data and knowledge bases. *IEEE Transactions on Knowledge and Data and Engineering*, 8, 353-371.
- Schneider, M. (2000). Metric operations on fuzzy spatial objects in databases. *8<sup>th</sup> ACM Symposium on Geographic Information Systems* (pp. 21-26). ACM Press.
- Servigne, S., Ubeda, T., Puricelli, A. & Laurini, R. (2000). A methodology for spatial consistency improvement of geographic databases. *GeoInformatica*, 4, 7-24.
- Shekhar, S., Coyle, M., Goyal, B., Liu, D.-R., & Sakar, S. (1997). Data models in geographic information systems. *Communications ACM*, 40, 103-111.
- Stock, O. (1997). *Spatial and temporal reasoning*. Dordrecht, The Netherlands: Kluwer Academic Library.
- Tryfona, N., & Egenhofer, M. (1997). Consistency among parts and aggregates: A computational model. *Transactions on GIS*, 1, 189-206.
- Usery, E. A. (1996). Conceptual framework and fuzzy set implementation for geographic features. In A. Frank, (Ed.), *Geographic objects with indeterminate boundaries GISDATA* (pp. 71-85). London: Taylor & Francis.
- Voisard, A., & David, B. (2002). A database perspective on geospatial data modeling. *IEEE Transactions on Knowledge and Data and Engineering*, 14, 226-246.

## KEY TERMS

**Inconsistency Tolerance:** It is the strategy that lets a system answers and processes data despite the fact that the databases are inconsistent.

**Multiple Spatial Representations:** Multiple representation levels encompass changes in geometric and topological structure of a digital object that occur with the changing resolution at which the object is encoded.

**Objects with Indeterminate Boundaries:** These are objects that lay in one of two categories: objects with sharp boundaries but whose position and shape are



unknown or cannot be measured exactly; or objects with not well-defined boundaries or for which it is useless to fix boundaries.

**Spatial Consistency:** It refers to the agreement between data representation and a model of space.

**Spatial-Integrity Constraints:** They refer to constraints that address properties with respect to a model of the space. They are usually classified into topological integrity constraints, semantic integrity constraints, and user-defined integrity constraints.

**Topological Consistency:** Two representations are consistent if they are topologically equivalent or if the

topological relations are valid under certain consistency rules.

**Topological Invariants:** These are properties that are invariant under topological transformations. Examples of topological invariants are the interior, boundary, and exterior of spatial objects.

**Topological Relevance:** Two representations are said to be topologically equivalent if one can be mapped into the other by a topological transformation of the real plane. Examples of topological transformations are rotation, scale change, translation, and symmetry.

# Converting a Legacy Database to Object-Oriented Database

**Reda Alhajj**

*University of Calgary, Canada*

**Faruk Polat**

*Middle East Technical University, Turkey*

## INTRODUCTION

We present an approach to transfer content of an existing conventional relational database to a corresponding existing object-oriented database. The major motivation is having organizations with two generations of information systems; the first is based on the relational model, and the second is based on the object-oriented model. This has several drawbacks. First, it is impossible to get unified global reports that involve information from the two databases without providing a wrapper that facilitates accessing one of the databases within the realm of the other. Second, organizations should keep professional staff familiar with the system. Finally, most of the people familiar with the conventional relational technology are willing to learn and move to the emerging object-oriented technology. Therefore, one appropriate solution is to transfer content of conventional relational databases into object-oriented databases; the latter are extensible by nature, hence, are more flexible to maintain. However, it is very difficult to extend and maintain a conventional relational database.

We initiated this study based on our previous research on object-oriented databases (Alhajj & Arkun, 1993; Alhajj & Elnagar, 1998; Alhajj & Polat, 1994, 1998); we also benefit from our research on database re-engineering (Alhajj, 1999; Alhajj & Polat, 1999). We developed a system and implemented a first prototype that takes characteristics of the two existing schemas as input and performs the required data transfer. For each of the two schemas, some minimum characteristics must be known to the system in order to be able to perform the transfer from tuples of the relational database into objects in the object-oriented database. We assume that all relations are in the third normal form, and the two schemas are consistent; that is, for every class, there is a corresponding relation. This consistency constraint necessitates that attributes with primitive domains in a certain class have their equivalent attributes in the corresponding relation, and values of non-primitive attributes in a class are determined

based on values of the corresponding foreign keys in the corresponding relation. Concerning the migrated data, it is necessary that consistency and integrity are maintained. Finally, the transfer process should not result in any information loss.

## BACKGROUND

There are several approaches described in the literature to facilitate accessing content of a relational database from an object-oriented application. DRIVER (Lebastard, 1995) proposes an object wrapper that allows relational database reusing. It uses relational database management systems as intelligent file managers and proposes object models on top of them. The user is expected to provide the mapping between the two schemas. Persistence (Agarwal, Keene & Keller, 1995; Keller, Agarwal & Jensen, 1993) is an application development tool that uses an automatic code generator to merge C++ applications with relational databases. The application object model is mapped to a relational schema in the underlying database. Therefore, object operations are transformed into relational operations and vice versa. The benefits and risks of the migration process are discussed in Keller and Turner (1995). The authors argue that storing data in a relational database and providing a wrapper to allow programming in an object programming language provides more benefit at significantly reduced risks and costs as compared with migrating to an object-oriented database. We argue that providing a wrapper adds a performance cost to be paid every time the data is processed because the mapping between objects and tuples is repeated dynamically in each session. But, the mapping cost is paid only once by migrating to an object-oriented database.

## MAIN THRUST

In this section, we present the basic necessary analysis that results in the information required to transfer the

data safely. First, we identify characteristics of the object-oriented schema. Then, we concentrate on characteristics of the relational schema. The last step leads to the equivalence between attributes in the two schemas. This information directs the system about the construction of objects out of tuples.

We investigate characteristics of the given object-oriented schema and construct the necessary tables. First, we present the basic terminology and definitions required to understand the analysis. We are mainly interested in the following class characteristics:

1.  $C_p(c)$ : list of the direct superclasses of class  $c$ .
2.  $L_{attributes}(c)$ : set of (not inherited) attributes locally defined in class  $c$ .
3.  $L_{instances}(c)$ : set of identifiers of objects added locally to class  $c$ .
4.  $W_{instances}(c)$ : extent of class  $c$ , that is, objects in  $L_{instances}(c)$  and in all direct and indirect subclasses of class  $c$ .
5. **OIDG**: identifier generator that holds the identifier to be granted to the next object to be added to  $L_{instances}(c)$ .

**Definition 1**

[Domain] Let  $c_1, c_2, \dots,$  and  $c_n$  be primitive and user defined classes, where primitive classes include reals, integers, strings, and so forth. The following are possible domains:

1.  $(a_1:c_1, a_2:c_2, \dots, a_n:c_n)$  is a tuple domain; a possible value is a tuple that consists of object identifiers selected from classes  $c_1, c_2, \dots,$  and  $c_n$ , respectively.
2.  $c_i, 1 \leq i \leq n$  is a domain; a possible value is an object identifier from class  $c_i$ .
3.  $d$  is a domain, which may be any of the two domains defined in 1 and 2; a possible value is a set of values from domain  $d$ .
4.  $[d]$  is a domain, which may be any of the two domains defined in 1 and 2; a possible value is a list of values from domain  $d$ .

**Example 1:** [Object-Oriented Schema]

**Person:**

$C_p(Person)=[]$   
 $L_{attributes}(Person)={SSN:integer, name:string, age:integer, sex:character, spouse:Person, nation:Country}$

**Country:**

$C_p(Country)=[]$   
 $L_{attributes}(Country)={Name:string, area:integer, population:integer}$

**Student:**

$C_p(Student)=[Person]$   
 $L_{attributes}(Student)={StudentID:integer, gpa:real, student\_n:Department, Takes:{(course:Course, grade:string)}}$

**Staff:**

$C_p(Staff)=[Person]$   
 $L_{attributes}(Staff)={StaffID:integer, salary:integer, works\_in:Department}$

**ResearchAssistant:**

$C_p(ResearchAssistant)=[Student, Staff]$   
 $L_{attributes}(ResearchAssistant)={}$

**Course:**

$C_p(Course)=[]$   
 $L_{attributes}(Course)={Code:integer, title:string, credits:integer, Prerequisite:{Course}}$

**Department:**

$C_p(Department)=[]$   
 $L_{attributes}(Department)={Name:string, head:Staff}$

**Secretary:**

$C_p(Secretary)=[Person]$   
 $L_{attributes}(Secretary)={words/mimute:integer, works\_in:Department}$

Related to the object-oriented schema, the analysis is based on the domain information summarized in:

ObjectAttributes(class name, attribute name, domain, Score)

Table 1. ObjectAttributes: A list of all attributes with non-primitive domains

Class Name	Attribute Name	Domain	Score
Person	spouse	Person	1
Person	nation	Country	1
Student	student_in	Department	1
Student	Takes	T <sub>1</sub>	2
Staff	works_in	Department	1
Course	prerequisite	Course	2
Department	head	Staff	1
Secretary	works_in	Department	1
T <sub>1</sub>	course	Course	1

This table includes information about all attributes with non-primitive domains in the object-oriented schema. The value of Score is 2 for tuple domains and multi-valued (set and list) domains; otherwise, it is 1. Table 1 is ObjectAttributes of the object-oriented schema in Example 1. Notice that each domain of the tuple type is assigned a short name that consists of the letter T suffixed with a consecutive non-decreasing number, starting with 1, as illustrated in the fourth row in Table 1. This way, it becomes trivial to identify attributes that appear within a tuple domain as illustrated in the last row. As the score is concerned, the value 2 appears in the fourth and sixth rows, which represent tuple and set domains, respectively.

With respect to the relational schema, it is necessary to identify primary and foreign keys of the given relations. Keys related information leads to all relationships that exist between relations in the relational schema. Each such relationship may correspond to an inheritance or a nesting relationship in the object-oriented schema. Consider the relational schema in Example 2, which will be referenced in the article to illustrate the presented approach.

**Example 2:** [Example Relational Schema]

- Person(SSN, name, city, age, sex, SpouseSSN, CountryName)
- Country(Name, area, population)
- Student(StudentID, gpa, PSSN, DName)
- Staff(StaffID, salary, PSSN, DeptName)
- ResearchAssistant(StudentID, StaffID)
- Course(Code, title, credits)
- Prerequisite(CourseCode, PreqCode)
- Takes(Code, StudentID, grade)
- Department(Name, HeadID)
- Secretary(SSN, words/mimute, DName)

The information required to feed the system with the necessary understanding about the given relational schema is summarized in Table 3.

*ForeignKeys(PK relation name, PK attribute name, FK relation name, FK attribute name, Link#, Score)*

Table 3 includes all pairs of attributes, such that the first attribute is part of a primary key, and the second attribute is part of a corresponding foreign key, a representative of the first attribute within any of the relations (including relation of the primary key itself). Link# classifies and differentiates attributes that constitute each foreign key. Foreign keys are numbered within each relation such that all attributes in the same foreign key are assigned the same value for Link#. The score is assigned in order to classify primary-foreign keys links as follows. A link that represents a subtyping is assigned the score 0, and all

Table 2. ForeignKeys: Attributes in primary keys and corresponding foreign keys

Primary Key attributes		Foreign Key attributes		Link#	Score
Relation Name	Attribute Name	Relation Name	Attribute Name		
Person	SSN	Student	PSSN	1	0
Person	SSN	Staff	PSSN	1	0
Person	SSN	Secretary	PSSN	1	0
Person	SSN	Person	SpouseSSN	1	1
Country	Name	Person	CountryName	2	1
Student	StudentID	ResearchAssistant	StudentID	1	0
Student	StudentID	Takes	StudentID	1	2
Staff	StaffID	ResearchAssistant	StaffID	2	0
Staff	StaffID	Department	HeadID	1	1
Course	Code	Prerequisite	CourseCode	1	2
Course	Code	Prerequisite	PreqCode	2	1
Course	Code	Takes	Code	2	1
Department	Name	Student	Dname	2	1
Department	Name	Staff	DeptName	2	1
Department	Name	Secretary	DName	2	1

Table 3. Equivalence: Object-oriented attributes with primitive domains and the corresponding relational attributes

Relation Name	Attribute Name	Class Name	Attribute Name
Person	SSN	Person	ssn
Person	name	Person	name
Person	age	Person	age
Person	sex	Person	sex
Country	Name	Country	name
Country	area	Country	area
Country	population	Country	population
Student	StudentID	Student	studentID
Student	gpa	Student	gpa
Staff	StaffID	Staff	staffID
Staff	salary	Staff	salary
Course	Code	Course	code
Course	title	Course	title
Course	credits	Course	credits
Department	Name	Department	name
Secretary	words/minute	Secretary	words/minute
Takes	grade	T1	grade

other links are assigned the score 1 or 2. Score 2 is assigned only to one of the links connected to a relation which represents a relationship. The latter link is selected based on the corresponding domain information (multi-valued or tuple domain) in the object-oriented schema, as illustrated next.

Table 2 is ForeignKeys of the relational schema in Example 2; the fourth tuple shows SpouseSSN as a foreign key within relation Person itself. The fourth and fifth rows in Table 2 represent two foreign keys within Person; their Link# is 1 and 2, respectively. Other values for Link# are assigned by the same way. As the Score is concerned, the score in the seventh row is 2 because Takes represents a relationship, and the corresponding tuple domain in the object-oriented schema is defined in class Student. Also, the value of Score in the tenth row is 2 because the corresponding attribute in class Course has a multi-valued domain.

Table 4. *EquivOfForKeys*: Object-oriented attributes with non-primitive domains and Link# of corresponding foreign keys in the relational schema

Class Name	Attribute Name	Relation Name	Link #
Person	spouse	Person	1
Person	nation	Person	2
Student	student_in	Student	2
Student	takes	Takes	1
Staff	works_in	Staff	2
Course	prerequisite	Prerequisite	1
Department	head	Department	1
Secretary	works_in	Secretary	2
T <sub>1</sub>	course	Takes	2

The basic goal to be achieved prior to data transfer is to decide on the destination of each data item to migrate from the relational database into the corresponding object-oriented database. The required information is summarized in the two tables.

*Equivalence*(relation name, relational attribute name, class name, object attribute name)

*EquivOfForKeys*(class name, object attribute name, relation name, Link#)

Equivalence keeps primitive attributes in the object-oriented schema and their corresponding attributes in the relational schema; *EquivOfForKeys* holds a tuple for each non-primitive attribute in the object-oriented schema to show its corresponding foreign key in the relational schema. A foreign key is referenced by its Link# from *ForeignKeys*. Tables 3 and 4 are, respectively, *Equivalence* and *EquivOfForKeys*, as the object-oriented schema in Example 1 and the corresponding relational schema in Example 2 are concerned. The first row in Table 4 indicates that attribute *spouse* from class *Person* corresponds to the foreign key that has Link#=1 in relation *Person*.

## THE TRANSFER MECHANISM

It is important to perform the transfer in two passes because non-primitive attributes get their values in terms of object identifiers that are not all known except after all objects are created. We have two algorithms, one for each pass. Algorithm 1 handles the first pass, where only identifiers are generated for all objects constructed out of tuples in the relational database, and the values of primitive attributes are assigned in each constructed

object. Due to sub-typing, multiple tuples distributed in different relations are used to construct a single object. So, classes in the inheritance hierarchy are considered by Algorithm 1 in order, starting with the leaves. During the process, all tuples that have been utilized in constructing objects are marked to be excluded from consideration when the time comes to construct objects in classes that correspond to their relations. An object has a corresponding tuple in relation *R* that corresponds to its class and in each of the direct and indirect super-relations of relation *R*. Finally, corresponding objects are constructed only for not marked tuples, that is, not related with other tuples.

After all object identifiers become known, Algorithm 2 performs the second pass to decide on the actual values of non-primitive attributes in each of the objects generated during the first pass. Classes are considered in top-down order during the second pass because it is necessary to determine the value of each attribute with non-primitive domain for all objects that expect a value for such attribute. The later set of objects covers the extent of the class in which the mentioned attribute is locally defined.

**Algorithm 1:** [Constructing objects: first pass]

**Input:** Equivalence table and relational database content.

**Output:** Objects are constructed, and values of attributes with primitive domains are assigned.

Consider classes in order, according to their position in the inheritance hierarchy, starting with leaves:

For every class *c* and its corresponding relation *R* do  
 For every tuple *r* in relation *R* do  
     If tuple *r* is not marked then  
         Locate and mark the tuple that joins with *r* in each of the super-relations of *R*, if any;  
         Construct in  $L_{instances}(R)$  an object with identifier *oid* to represent tuple *r* such that, related to object *oid*:

- Based on the information present in *Equivalence*, attributes with primitive domains in  $W_{attributes}(R)$  get their values from the corresponding tuple in *R* and the already located related tuples in the super-relations of *R*;
- Assign *nil* to attributes with complex domains in  $W_{attributes}(R)$ ;

**Algorithm 2:** [Constructing objects: second pass]



**Input:** The three tables *EquivOfForKeys*, *Equivalence* and *ForeignKeys*, and the relational database.

**Output:** Values of the attributes with non-primitive domains are assigned for all objects constructed by Algorithm 1.

Consider classes in breadth first order, starting with direct subclasses of the root:

```

For every class c do
  For every attribute  $a_o \in L_{attributes}(c)$  with non-primitive domain do
    Let ( $c, a_o, R, Link\#$ ) be corresponding tuple in EquivOfForKeys.
    Locate all tuples ( $R', a_r, R, a_r, Link\#, Score$ ) in ForeignKeys.
    Let  $pk'$  be primary key of  $r'$  and  $fk'$  be the corresponding foreign key in  $R$ .
    Locate all tuples in ForeignKeys, where  $R$  appears in the third column, and let  $n$  be the maximum value of Link# in the located tuples.
    If at least one of the located tuples has  $Score=2$  then
      If  $n=2$  and  $R$  does not appear in the first column of Equivalence then
        Let  $R''$  be the other relation that appears in the first column of the located tuples
        Let  $pk''$  be primary key of  $R''$  and  $fk''$  be the corresponding foreign key in  $R$ .
        For every tuple  $r'$  in relation  $R'$  do
          Let  $oid \in W_{instances}(c)$  be the identifier of the object that corresponds to tuple  $r'$ ;
          Locate in relation  $R$ , every tuple  $r$  that joins with tuple  $r'$  based on the equality of  $fk'$  and  $pk'$ ;
          Locate in relation  $R''$ , every tuple  $r''$  that joins with tuples  $r$  based on the equality of  $fk''$  and  $pk''$ ;
          Assign identifiers of the objects that correspond to tuples  $r''$  as the value of attribute  $a_o$  in object  $oid$ ;
        Else
          Let  $R_i''$ ,  $i=1, n-1$ , be the other relations that appear in the first column of the located tuples
          Let  $pk_i''$  be primary key of  $R_i''$  and  $fk_i''$  be the corresponding foreign key in  $R$ ,  $i=1, n-1$ 
          For every tuple  $r'$  in relation  $R'$  do
            Let  $oid \in W_{instances}(c)$  be the identifier of the object that correspond to tuple  $r'$ ;
            Locate in relation  $R$ , every tuple  $r$  that joins with tuple  $r'$  based on the equality of  $fk'$  and  $pk'$ ;
            For  $i:=1$  to  $n-1$  do
              Locate in relation  $R_i''$ , every tuple  $r_i''$  that joins with tuples  $r$  based on the equality of  $fk_i''$  and  $pk_i''$ ;
            Construct new tuples by using identifiers of the objects that correspond to tuples  $r_i''$  and values of non-foreign key attributes in tuples  $r$ .
            Assign the constructed tuples as the value of attribute  $a_o$  in object  $oid$ ;

```

```

Else
  For every tuple  $r$  in relation  $R$  do
    Let  $oid \in W_{instances}(c)$  be the identifier of the object that correspond to tuple  $r$ ;
    Locate in relation  $R'$ , every tuple  $r'$  that joins with tuple  $r$  based on the equality of  $fk'$  and  $pk'$ ;
    Assign the identifiers of the objects that correspond to tuples  $r'$  as the value of attribute  $a_o$  in object  $oid$ ;

```

## FUTURE TRENDS

XML is considered the main standard for data exchange, so a lot of research is being done for transforming into XML format data stored in the existing other data models. The current research in the field mainly covers relational model; we have already achieved some progress in this direction; and our current work is mainly based on the reverse engineering process described in this article. We derive the entity-relationship model and then transform it into XML (Lee et al., 2002; Lo, Alhadj & Barker, 2004; Wang et al., 2004). We argue that other models should also receive equal attention. It is also possible to study different aspects of the obtained XML schema, including query processing and optimization, view maintenance, and schema evolution.

## CONCLUSION

In this article, we developed a novel approach to transfer data from the relational database into a corresponding object-oriented database. To achieve the target without violating consistency and integrity, we analyzed the two database schemas. The analysis led to several tables that summarize characteristics of the two schemas, as well as the equivalence between their attributes. Then two algorithms were presented. The first algorithm constructs objects by assigning their identifiers and the values of their primitive attributes. The second algorithm decides on the values of non-primitive attributes in each constructed object. This way, contents of the two databases are unified, giving the opportunity to process the whole information at once with more flexibility to produce global reports that cover information from different databases. As a result, this approach has the advantage that once the data has been ported, no dynamic data mapping is required between tuples and objects, thus removing the run-time overhead. Finally, being aware of the huge sizes of the data to be transferred, we are planning to improve the performance by parallelizing the two algorithms. We expect the marking of tuples to play an important role in the parallelization process.

## REFERENCES

Agarwal, S., Keene, C., & Keller, A.M. (1995). Architecting object applications for high performance with relational databases. *Proceedings of the OOPSLA Workshop on Object Database Behavior, Benchmarks, and Performance*, Austin, Texas.

Alhajj, R. (1999). Documenting legacy relational databases: Mining the entity-relationship model. *Proceedings of the REIS*, Paris.

Alhajj, R., & Arkun, M.E. (1993). A query model for object-oriented database systems. *Proceedings of the IEEE-ICDE* (pp. 163-172).

Alhajj, R., & Elnagar, A. (1998). Incremental materialization of object-oriented views. *DKE*, 29(2), 121-145.

Alhajj, R., & Polat, F. (1998). Proper handling of query results towards maximizing reusability in object-oriented databases. *Information Sciences: An International Journal*, 107(1-4), 247-272.

Alhajj, R., & Polat, F. (1994). Closure maintenance in an object-oriented query model. *Proceedings of ACM-CIKM* (pp. 72-79).

Alhajj, R., & Polat, F. (1999). Database reverse engineering. *Proceedings of the ISICIS*, Kusadasi.

Keller, A.M., Agarwal, S., & Jensen, R. (1993). Enabling the integration of object applications with relational databases. *Proceedings of the ACM-SIGMOD*, Washington, DC.

Keller, A.M., & Turner, P. (1995). Migrating to object data management. *Proceedings of the OOPSLA Workshop on Legacy Systems and Object Technology*, Austin, Texas.

Lebastard, F. (1995). Is an object layer on a relational database more attractive than an object database? *Proceedings of the Workshop on Reasoning about Structured Objects: Knowledge Representation Meets Databases*, Bielefeld.

Lee, D., et al. (2002). NeT and CoT: Translating relational schemas to XML schemas using semantic constraints. *Proceedings of the ACM CIKM*, VA.

Lo, A., Alhajj, R., & Barker, K. (2004). Flexible user interface for converting relational data into XML. *Proceedings of the International Conference on Flexible Query Answering Systems*, France.

Wang, C., Lo, A., Alhajj, R., & Barker, K. (2004). Converting legacy relational database into XML database through reserve engineering. *Proceedings of the International Conference on Enterprise Information Systems*, Portugal.

## KEY TERMS

**Candidate Key:** Minimum set of attributes that uniquely identify each tuple of a given relation. One candidate key is selected as primary key.

**Class:** A collection of objects that have the same behavior and state definition.

**Database Reverse Engineering:** The process of analyzing an existing database to identify its components and their interrelationships and to create representations of another data model.

**Entity-Relationship Diagram:** A graphical representation of the entities and the relationships between them. Entity relationship diagrams are a useful medium to achieve a common understanding of data among users and application developers.

**Foreign Key:** A set of one or more attributes that correspond to a primary key in order to simulate a relationship. It links two tables.

**Object:** A data structure that encapsulates behavior (operations) and data (state).

**Object-Oriented Database:** A database in which the operations carried out on information items (data objects) are considered part of their definition.

**Relational Database:** A collection of data items organized as a set of formally-described tables from which data can be accessed or reassembled in many different ways without having to reorganize the database tables.

# Data Dissemination

**Ludger Fiege**

*Technische Universität Darmstadt, Germany*

## INTRODUCTION

One of the driving factors in IT development is the availability of cheap and efficient network technologies. The Internet is no longer used only as a medium for personal communication. Organizations start utilizing this technology to link existing applications and to create new ones. While traditionally systems were designed to respond to interactive user requests, they are now more and more aiming at autonomous, distributed data processing. Systems are connected and instantly react to changes to improve their functionality and utility (cf. *zero latency enterprises*). Mobile systems and other volatile configurations demand reactions to continuous changes; finance applications must be notified of price fluctuations; supply chain management must observe stock level changes; and information retrieval applications must forward new content (Banavar, Chandra, Strom, & Sturman, 1999; Gray, 2004).

The focus when dealing with the delivery of data and services is changing, moving from a stationary world to one that is in a state of flux. Traditionally, data and services have been viewed as being stationary in a collection of objects or databases, with inquiries directed to them in a request/reply mode of interaction. This concept has led to client/server architectures that emphasize explicit delegation of functionality, where processes access remote functionality to accomplish their own goal. Remote procedure calls (RPC) and derivative techniques are classic examples (Birrell & Nelson, 1984; Mullender, 1993); even the incipient Web services mainly rely on sending requests with the Simple Object Access Protocol (SOAP; Alonso, Casati, Kuno, & Machiraju, 2003). These techniques deliberately draw from a successful history of engineering experience, their principles are well understood, and they have been an appropriate choice for many well-defined problems.

In the context of dynamic or large-scale applications, however, request/reply has serious restrictions. The direct and often synchronous communication between clients and servers enforces a tight coupling of the communicating parties and impairs scalability (Franklin & Zdonik, 1998). Clients poll remote data sources, and they have to trade resource usage for data accuracy, especially in chains of dependent servers. Unnecessary requests due to short polling intervals waste resources, whereas long intervals increase update latency. The obvious need for

asynchronous and decoupled operation has led to various extensions of existing middleware standards. For instance, CORBA and Java 2 Enterprise Edition (J2EE) were extended with asynchronous invocation methods and notification services (Object Management Group [OMG], 1999; Schmidt & Vinoski, 1999; Sun Microsystems, 2002). Database research, software engineering, and coordination theory corroborate the advantages of loosely coupled interaction (Cilia, Bornhövd, & Buchmann, 2001; Papadopoulos & Arbab, 1998; Sullivan & Notkin, 1992). The following presents a classification of communication paradigms and technologies to outline their fundamental characteristics.

## COMMUNICATION PARADIGMS

Irrespective of the actual technology used for communication, different modes of interaction can be distinguished that determine the way interdependencies between processes are established. Four models are differentiated by two dimensions (see Table 1; Fiege, Mühl, & Gärtner, 2003). The first attribute, *initiator*, describes whether the consumer or the provider of data initiates the interaction, where the former depends on data or functionality provided by the latter. The second attribute, *addressing*, distinguishes whether the addressee of the interaction is known or unknown, i.e., whether peer(s) are addressed directly or indirectly. Any interaction between a set of processes can be classified according to these models. Even though interaction may show more nuances in practice, the models are complete in the sense that they essentially cover all major paradigms.

*Request/reply* is the most widely used interaction model. Any kind of remote procedure call or client/server interaction belongs to this class. The initiator is the consumer (i.e., client) that requests data and/or function-

*Table 1. Four models of interaction*

<b>Initiator</b> <b>Addressing</b>	<b>Consumer</b>	<b>Provider</b>
<b>Direct</b>	Request/Reply	Callback
<b>Indirect</b>	Anonymous Request/Reply	Event-based

ality from the provider (i.e., server). It expects data to be delivered back or relies on a specific task to be done. The provider is directly addressed, its identity is known, and the caller is able to incorporate information about the callee into his own state and processing, resulting in a tight coupling of the cooperating entities. Replies are mandatory in this model unless the system and failure model excludes errors.

*Anonymous request/reply* does not specify the provider that should process the request. Instead, requests are delivered to an arbitrary, possibly dynamically determined set of providers. The consumer does not know the identity of the recipient(s) a priori, yet it expects at least one reply. This model is eligible when multiple providers are available that are selected at runtime by the communication subsystem, e.g., according to the specific parameters contained in a request or according to load balancing policies.

In the *callback* mode consumers register at a specific, known provider their interest to be notified whenever an observed condition becomes true. The provider repeatedly evaluates the condition and if necessary calls back the registered process. This kind of interaction is employed in the well-known *observer* design pattern (Gamma, Helm, Johnson, & Vlissides, 1995). The provider is responsible for maintaining a list of registered consumers. If multiple callback providers are of interest, a consumer must register at each of them separately. The identity of peers is known and must be managed by the application.

The *event-based* interaction model has characteristics inverse to the request/reply model. The initiator of communication is the provider of data, that is, the producer of notifications. Notifications are not addressed to any specific set of recipients; consumers rather issue subscriptions that describe what kind of data they are interested in. A notification is delivered to a consumer if it matches one of its subscriptions. Providers are not aware of consumers and vice versa. An intermediary messaging service conveys the notifications. The essential characteristic of this model is that without knowing consumers of notifications, producers send information only about their own state, precluding any assumptions on consumer functionality. The overall functionality is not encoded in the processes but determined implicitly by their composition and interaction.

Request/reply and event-based interaction are characterized by the simplicity of the former and the flexibility of the latter. Request/reply is easy to handle, implement, and understand. It corresponds to the imperative nature of the client/server paradigm and of common programming languages. However, this model has three principal drawbacks:

1. Architecture lock.
2. Point-to-point communication limits scalability.
3. Polling limits accuracy in automated computations.

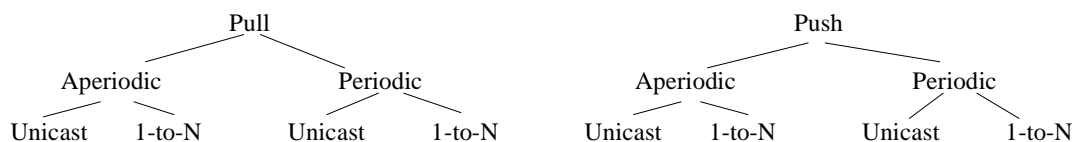
The architecture of the system is locked up in application code, and changes to an existing configuration are difficult to deploy at runtime (Garlan & Shaw, 1993; Papadopoulos & Arbab, 1998). The explicit point-to-point communication relies on the presence of specific servers and requires that peers are online simultaneously. This is a severe restriction considering the increasing interconnection of IT systems which reside in different organizational domains. Obviously, bandwidth consumption is another limiting factor of point-to-point communication. Finally, when replacing interactive with automated data processing, computation becomes *information-driven*. It is no longer initiated on request, but when new input data is available.

Databases are affected in two ways. First, data stores are no longer the important central aspect of future architectures. Traditional database functionality fulfills helper functions. Second, and more importantly, database research contributes to the ongoing work on data dissemination systems.

## TECHNIQUES

Generally, there is no best implementation technique for a certain model of interaction. The technique must be chosen in view of the deployment environment, the required quality of service, and the need for flexibility and scalability. A classification of data dissemination techniques was presented by Franklin and Zdonik in 1998 (see Figure 1). They distinguish client pull vs. server push, periodic vs. aperiodic, and unicast vs. 1-to-N (multicast) delivery. These alternatives may even be mixed in one

Figure 1. Classification of data delivery





infrastructure consisting of data sources, clients, and a number of brokers (Deolasee, Katkar, Panchbudhe, Ramamritham, & Shenoy, 2001).

*Aperiodic pull* with unicast comprises traditional remote procedure calls (Birrell & Nelson, 1984). Note that this classification is about communication techniques, and while RPCs typically convey requests, they may also be used to transport notifications within a messaging middleware layer. In fact, point-to-point communication is often used to distribute notifications (Carzaniga, Rosenblum, & Wolf., 2001) because of the limited availability of multicast networks. If used with a 1-to-N delivery (e.g., IP Multicast) multiple clients can “snoop” the results (Acharya, Franklin, & Zdonik, 1997; Liebig, Cilia, Betz, & Buchmann, 2000). Interestingly, the emerging Web services mainly stick to this model and offer only initial push support on top of the Simple Mail Transport Protocol (Klensin, 2001).

*Periodic pull* is unfortunately often encountered in automated computations. Many of the first push systems on the Web suffered from this implementation of event-based interaction (Franklin & Zdonik, 1998). In CORBA the notification service explicitly allows a pull-based delivery of messages (OMG, 1999).

*Aperiodic push* mainly consists of publish/subscribe messaging systems (Carzaniga et al., 2001; Oki, Pfluegl, Siegel, & Skeen, 1993; Yan & Garcia-Molina, 1999), which may be implemented with unicast and multicast delivery. They send published messages to all subscribers that have registered a matching filter. Note that publish/subscribe may implement the event-based as well as the anonymous request/reply model.

*Periodic push* is a special case of aperiodic push and comprises the same techniques, but there are also specialized techniques available such as broadcast disks (Acharya et al., 1997).

A periodic push approaches have received much attention recently. Publish/subscribe inverts the traditional data-query processing, and thus it can draw from database research. Changes that are to be published must be searched and detected (e.g., Diao, Altinel, Franklin, Zhang, & Fischer, 2003). Efficient indexing (Yan & Garcia-Molina, 1994) and query handling (Madden & Franklin, 2002) is needed for message processing. Active database systems investigate event composition, which is of increasing importance once the subscription operators are getting more expressive (Liebig, Cilia, & Buchmann, 1999). Finally, in-network processing of data raises issues related to flow control, caching (Deolasee et al., 2001; Intanagonwiwat, Govindan, Estrin, Heidemann, & Silva, 2003), message substrates, and data security and access control (Bertino, Ferrari, & Pitoura, 2001).

## CONCLUSION

Networking continues to shape the architecture of IT systems. Existing applications are interconnected to increase their utility, and new applications are created, driven by emerging new standards like Web services and by the need to automate data processing. The presented classification of communication paradigms and techniques helps to differentiate the characteristics of different forms of data dissemination. The dominant request/reply mode of interaction has inherent limitations in large networks, and it is often replaced by publish/subscribe approaches. Database research can contribute efficient subscription handling, filter operators, and data caching.

## REFERENCES

- Acharya, S., Franklin, M., & Zdonik, S. (1997). Balancing push and pull for data broadcast. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 183-194).
- Alonso, G., Casati, F., Kuno, H., & Machiraju, V. (2003). *Web services*. Springer-Verlag.
- Banavar, G., Chandra, T. D., Strom, R. E., & Sturman, D. C. (1999). A case for message oriented middleware. In *Lecture notes in computer science: Vol. 1693. 13th International Symposium on Distributed Computing, DISC'99* (pp. 1-17). Springer-Verlag.
- Bertino, E., Ferrari, E., & Pitoura, E. (2001). An access control mechanism for large scale data dissemination systems. In *International Workshop on Research Issues in Data Engineering: Document Management (RIDE-DM)* (pp. 43-50).
- Birrell, A., & Nelson, B. (1984). Implementing remote procedure calls. *ACM Transactions on Computer Systems*, 2(1), 39-59.
- Carzaniga, A., Rosenblum, D. S., & Wolf, A. L. (2001). Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3), 332-383.
- Cilia, A., Bornhövd, C., & Buchmann, A. P. (2001). Moving active functionality from centralized to open distributed heterogeneous environments. In *Lecture notes in computer science: Vol. 2172. Proceedings of the 6th International Conference on Cooperative Information Systems, CoopIS 2001* (pp. 195-210). Springer-Verlag.



- Deolasee, P., Katkar, A., Panchbudhe, A., Ramamritham, K., & Shenoy, P. (2001). Adaptive push-pull: Disseminating dynamic Web data. *International World Wide Web Conference*, (pp. 265-274).
- Diao, Y., Altinel, M., Franklin, M. J., Zhang, H., & Fischer, P. (2003). Path sharing and predicate evaluation for high-performance XML filtering. *ACM Transactions on Database Systems*, 28(4), 467-516.
- Fiege, L., Mühl, G., & Gärtner, F. C. (2003). Modular event-based systems. *The Knowledge Engineering Review*, 17(4), 359-388.
- Franklin, M. J., & Zdonik, S. B. (1998). "Data in your face": Push technology in perspective. In *Proceedings of ACM International Conference on Management of Data, SIGMOD'98* (pp. 516-519).
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns*. Addison-Wesley.
- Garlan, D., & Shaw, M. (1993). An introduction to software architecture. In V. Ambriola & G. Tortora (Eds.), *Advances in software engineering and knowledge engineering* (Vol. 1, pp. 1-40.) World Scientific.
- Gray, J. (2004). The next database revolution. In *ACM International Conference on Management of Data, SIGMOD'04* (pp. 1-4).
- Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J., & Silva, F. (2003). Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, 11(1), 2-16.
- Klensin, J. (2001). *Simple Mail Transfer Protocol* (RFC 2821).
- Liebig, C., Cilia, M., Betz, M., & Buchmann, A. (2000). A publish/subscribe CORBA persistent state service prototype. In *Lecture notes in computer science: Vol. 1795. IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing, Middleware 2000*. Springer-Verlag.
- Liebig, C., Cilia, M., & Buchmann, A. (1999). Event composition in time-dependent distributed systems. In *Proceedings of the 4th International Conference on Cooperative Information Systems, CoopIS'99* (pp. 70-78).
- Madden, S., & Franklin, M. J. (2002). Fjording the stream: An architecture for queries over streaming sensor data. In *International Conference on Data Engineering, ICDE'02* (pp. 555-566).
- Mullender, S. (1993). *Distributed systems* (2nd ed.). Addison-Wesley.
- Object Management Group. (1999). *CORBA notification service* (telecom/99-07-01).
- Oki, B., Pfluegl, M., Siegel, A., & Skeen, D. (1993). The information bus—An architecture for extensible distributed systems. In *Proceedings of the 14th Symposium on Operating Systems Principles* (pp. 58-68).
- Papadopoulos, G. A., & Arbab, F. (1998). Coordination models and languages. In M. Zelkowitz (Ed.), *Advances in computers: Vol. 46. The engineering of large systems*. Academic Press.
- Schmidt, D. C., & Vinoski, S. (1999). Programming asynchronous method invocations with CORBA messaging. *C++ Report*, 11(2).
- Sullivan, K. J., & Notkin, D. (1992). Reconciling environment integration and software evolution. *ACM Transactions of Software Engineering and Methodology*, 1(3), 229-269.
- Sun Microsystems. (2002). *Java Message Service Specification 1.1*.
- Yan, T., & Garcia-Molina, H. (1994). Index structures for selective dissemination of information under the Boolean model. *ACM Transactions on Database Systems*, 19(2).
- Yan, T., & Garcia-Molina, H. (1999). The SIFT information dissemination system. *ACM Transactions on Database Systems*, 24(4).

## KEY TERMS

**Broadcast Disks:** Distribute data according to a pre-defined schedule so that interested clients can pick up the data from the schedule.

**Client/Server:** The architecture built on top of RPC, commonly used in business applications.

**Content Distribution Networks:** The general term encompassing any technology for wide-area distribution of content.

**Data Streams:** Opposite to the stationary view of databases, data streams focus on the continuous processing of newly arriving data, i.e., joining and comparing of streams.

**Events:** In contrast to request/reply, producers send unsolicited notifications about events they observe. Instead of delivering notifications directed to consumers, they are first sent to an intermediary service. This service delivers the notifications according to the subscriptions consumers have issued.

## **Data Dissemination**

**Messaging:** The general means of unidirectional data transmission. Messages may contain requests, responses or notifications.

**Push/Pull:** Distinguishes who initiates data transmission, the producer or the consumer of data.

**RPC:** Implements the classic request/reply communication paradigm. Clients directly communicate with specific servers and ask for data or functionality.

**D**

# Data Model Versioning and Database Evolution

**Hassina Bounif**

*Swiss Federal Institute of Technology, Switzerland*

## INTRODUCTION

In the field of computer science, we are currently facing a major problem designing models that evolve over time. This holds in particular for the case of databases: Their data models need to evolve, but their evolution is difficult. User requirements are now changing much faster than before for several reasons, among them the changing perception of the real world and the development of new technologies. Databases show little flexibility in terms of supporting changes in the organization of their schemas and data. Database evolution approaches maintain current populated data and software application functionalities when changing database schema. Data model versioning is one of these chosen approaches used to resolve the evolution of conventional and nonconventional databases such as temporal and spatial-temporal databases. This article provides some background on database evolution and versioning technique fields. It presents the unresolved issues as well.

## BACKGROUND

A major problem in computer science is how to elaborate designs that can evolve over time in order to avoid having to undergo redesign from scratch. Database designs are a typical example where evolution is both necessary and difficult to perform. Changes to database schema have to be enabled while maintaining the currently stored data and the software applications using the data in the database. One of the existing approaches used to resolve difficulties in database evolution is data model versioning. These techniques are the main focus of this article.

To avoid misinterpretation, we first recall the definition of the meaning of the basic concepts: schema versioning, schema evolution, schema modification, and schema integration. These concepts are all linked to the database evolution domain at different levels, but according to most of the research focusing on database evolution (Roddick, 1995), there is a consistent distinction between them.

- **Schema Versioning** means either creating from the initial maintained database schema or deriving from

the current maintained version different schemas called versions, which allow the querying of all data associated with all the versions as well as the initial schema according to user or application preferences. Partial schema versioning allows data updates through references to the current schema, whereas full schema versioning allows viewing and update of all present data through user-definable version interfaces (Roddick, 1995).

- **Schema Evolution** means modifying a schema within a populated database without loss of existing data. When carrying out an evolution operation, there are two problems that should be considered: on one hand, the semantics of change, i.e., their effects on the schema; and, on the other hand, change propagation, which means the propagation of schema changes to the underlying existing instances (Peters & Özsu, 1997). Schema evolution is considered a particular case of schema versioning in which only the last version is retained (Roddick, 1995).
- **Schema Modification** means making changes in the schema of a populated database: The old schema and its corresponding data are replaced by the new schema and its new data. This may lead to loss of information (Roddick, 1995).
- **Schema Integration** means combining or adding schemas of existing or proposed databases with or to a global unified schema that merges their structural and functional properties. Schema integration occurs in two contexts: (1) view integration for one database schema and (2) database integration in distributed database management, where a global schema represents a virtual view of all database schemas taken from distributed database environments. This environment could be homogenous, i.e., dealing with schemas of databases having the same data model and identical DBMSs (database management systems), or heterogeneous, which deals with a variety of data models and DBMSs. This second environment leads to what is called federated database schemas. Some research works consider schema integration a particular case of schema evolution in which the integration of two or more schemas takes place by choosing one and applying the facts in the others (Roddick, 1995).

This article is organized as follows. The next section reviews some background related to database evolution and focuses on the versioning data model. In the Future Trends section, we present some emerging trends. The final section gives the conclusion and presents some research directions.

## MAIN THRUST

### Versioning Methods and Data Conversion Mechanisms

Various mechanisms are used to manage versioning/evolution of databases according to the data model, the nature of the field of the applications, as well as the priorities adopted (Munch, 1995).

- **Revisions (Sequential):** consists of making each new version a modification of the most recent one. At the end, the versions sequentially form a single linked list called a *revision chain* (Munch, 1995). Each version in the chain represents an evolution of the previous one. This method is used for versioning schemas. This technique is not very effective because many copies of the database are created as schema versions. It is adopted by the Orion System (Kim & Chou, 1988), in which complete database schemas are created and versioned.
- **Variants (Parallel):** means changing the relationships for revision from one-to-one to many-to-one, so that many versions may have the same relationship with a common “ancestor” (Munch, 1995). A variant does not replace another, as a revision does, but is instead an alternative to the current version. This versioning method is adapted differently from one database to another depending on the unit (a schema, a type, or a view) to be versioned. For instance, the Encore System (Skarra & Zdonik, 1987) uses type versioning while the Frandole2 system (Andany, Leonard, & Palisser, 1991) uses view versioning. Revisions and variants are usually combined into a common structure named the *version graph*.
- **Merging:** consists of combining two or more variants into one (Munch, 1995).
- **Change Sets:** In this method, instead of having each object individually versioned, the user collects a set of changes to any number of objects and registers them as one unit of change to the database. This collection is termed a change set, or cset.

- **Attribution:** Used to distinguish versions via values of certain attributes, for example, a date or a status.

Other authors (Awais, 2003; Connolly & Begg, 2002) further categorize versions. A version can be transient, working, or released.

- **Transient versions:** A transient version is considered unstable and can be updated or deleted. It can also be created from scratch by checking out a released version from a public database or by deriving it from a working or transient version in a private database. In the latter case, the base transient version is promoted to a working version. Transient versions are stored in the creator’s private workspace (Connolly & Begg, 2002).
- **Working versions:** A working version is considered stable and cannot be updated, but it can be deleted by its creator. It is stored in the creator’s private workspace (Connolly & Begg, 2002).
- **Released versions:** A released version is considered stable and cannot be updated or deleted. It is stored in a public database by checking in a working version from a private database (Connolly & Begg, 2002).

With respect to data, there are three main data conversion mechanisms that are employed with these versioning methods: (1) Data is simply coerced into the new format; (2) Data is converted by lazy mechanisms when required; in this mechanism, there is no need to consider unneeded data, and schema changes take place rapidly when change occurs; and (3) Conversion interfaces are used to access data that never undergo physical conversion (Roddick, 1995).

### Versioning and Data Models

The unit to be versioned must have (1) a unique and immutable identity and (2) an internal structure. These two criteria have some consequences on the possibilities of versioning within different data models. Object-oriented or ER-based data models clearly fulfill these two requirements (Munch, 1995). The adoption of an object-oriented data model is the most common choice cited in the literature (Grandi & Mandreoli, 2003) on schema evolution through schema versioning. The relational model has also been studied extensively. In fact, some problems exist. Tuples, for instance, are simply values which fulfill the requirements for non-atomicity, but they have no useful ID. A user-defined key cannot be used since the

tuple may be updated and could suddenly appear as another. A complete relation table can, of course, be treated as an object with an ID, but this corresponds more to a type than to a normal object. Solutions for extended relational models have been proposed, such as the RM/T model, which introduces system-defined, unique, and immutable keys.

The management of versions is done with either schema versions or object versions.

- **Schema Versions:** A version of a schema consists of a triplet such as  $\langle S, N, v \rangle$ , where  $\langle S, N \rangle$  are an identification of the schema (S) and the version value (N). The value of the schema version, v, is a set of classes connected by inheritance relationships and references relationships such as composition.
- **Object Versions:** An object is associated with a set of versions. Under schema versions, associated objects belong only to one class. Representation and behavior of an object under a class constitute an object version under this class. An object version is a triplet such as  $\langle O, N, v \rangle$ , in which O is an object identifier, N, an integer, and v, the value of the version of an object. Object versions are identified by the couple  $\langle O, N \rangle$ . Generally, in models, an object has at least one version under a class; N is that of the corresponding class. Value v is defined in the type of the associated class. Three operations associated with the concept of object version are: creation of the first object version, deletion of object version, and derivation of a new object version.

### Transparency of Versioning

The management of schema versions or object versions is clear to the user, who does not have to constantly worry about the possible occurrence of versions. In order to gain non-versioned access to data stored in a versioned database, there are several alternative solutions (Munch, 1995).

1. Data is checked out to a workspace, where applications can access them freely, and then checked in to the database when they are no longer being used. This technique is most often used when dealing with files.
2. A layer is built on top of the versioned database which provides a standard interface and hides the details of the database, as well as the versioning. This technique is useful also for non-versioned databases.
3. Version selection is not part of the normal access functions of the database. Versioning is set up at the start and possibly the end of a change or a long transaction.

There are two main version storage strategies, which are:

- **Versioning-by-copy:** A new version of an object is implemented by making a copy of the object, assigning the copy a version number, and updating references to its predecessors. This technique consumes more disk space but provides a form of backup (Awais, 2003).
- **Versioning-by-difference:** It is also known as delta-versioning and requires that the differences between the new version and its predecessors be kept. Although this technique saves the disk space, it increases overheads since the DBMS needs to rebuild the particular version each time it has to be accessed (Awais, 2003).

### Versioned Database Functionality

The most fundamental operations that a DBMS has to provide to allow the versioning process are (Munch, 1995):

- creating new versions;
- accessing specific versions through a version selection mechanism;
- adding user-defined names or identifiers to versions;
- deleting versions (some restrictions are made on which versions are allowed to be deleted, if any at all);
- maintaining special relationships between versions, e.g., “revision\_of” or “variant\_of”;
- changing existing versions if possible;
- “freezing” certain versions so that they cannot be changed, if change is possible;
- merging variants, automatically and/or manually;
- attaching status values or other attributes to versions.

### Example of a Database Management System with Versioning

Orion (Kim & Chou, 1988) supports schema versions. It introduces invariants and rules to describe the schema evolution in OODBs (object-oriented databases). Orion defines five invariants and 12 rules for maintaining the invariants. The schema changes allowed are classified into three categories:

- Changes to the contents of a class;
- Changes to the inheritance relationship between classes;



- Changes to the class as a whole (adding and dropping entire classes)

In the following, some of these changes are described.

1. **Add an attribute V to a class C:** If V causes a name conflict with an inherited attribute, V will override the inherited attribute. If the old attribute was locally defined in C, it is replaced by the new definition. Existing instances to which V is added receive the null value.
2. **Drop an attribute V from a class C** and subclasses of C that inherited it. Existing instances of these classes lose their values for V. If C or any of its subclasses has other superclasses that have attributes of the same name then it inherits one of them.
3. **Make a class S a superclass of a class C:** The addition of a new edge from S to C must not introduce a cycle in the class lattice. C and its subclasses inherit attributes and methods from S.
4. **Add a new class C:** All attributes and methods from all superclasses specified for C are inherited unless there are conflicts. If no superclasses of C are specified, the system-defined root of the class lattice (in this case, CLASS) becomes the superclass of C.
5. **Drop an existing class C:** All edges from C to its subclasses are dropped. This means that the subclasses will lose all the attributes and methods they inherited from C. Then all edges from the superclasses of C to C are removed. Finally, the definition of C is dropped and C is removed from the lattice. Note that the subclasses of C continue to exist.

These are some rules for versioning of schemas.

- **Schema-Version Capability Rule:** A schema version may either be transient or working. A transient version may be updated or deleted, and it may be promoted to a working schema version at any time. A working schema version, on the other hand, cannot be updated. Further, a working schema version can be demoted to a transient schema version if it has no child schema version.
- **Direct-Access-Scope Update Rule:** The access scope of a schema version is non-updatable under that version if any schema version has been derived from it unless each of the derived schemas has blocked the inheritance from its parent or alternatively has left the scope of its parent.

## FUTURE TRENDS

Current research on the evolution of information systems and particularly on versioning databases is investigating ways of refining the versioning model by proposing new combined versioning approaches that aim to reduce the number of schema versions to a minimum in order to reduce the database access time and the database space consumption. An example can be seen in the framework based on the semantic approach that defines reasoning tasks in the field of description logic for evolving database schemas (Franconi, Grandi, & Mandreoli, 2000; Grandi & Mandreoli, 2003). Researchers are also concentrating on other new domains that seem more promising and similar to their research domain, such as:

### Evolution Based on Ontology Approach

Ontology evolution is not similar to database evolution even though many issues related to ontology evolution are exactly the same as those in schema evolution (Noy & Klein, 2002). For example, database researchers distinguish between schema evolution and schema versioning, while, for ontologies, there is no distinction between these two concepts. However, ontologies are used to resolve database evolution issues, as in works that present ontology of change (Lammari, Akoka, & Comyn-Wattiau, 2003). This ontology is a semantic network that classifies an evolution or a change into one or more categories depending on the significance of the change, its semantics, and the information and application dependency.

## CONCLUSION

Although the subject of database evolution has been extensively addressed in the past, for relational and object-oriented data models in temporal and other variants of databases, a variety of techniques has been proposed to execute change in the smoothest way to maintain data and application functionalities. Many research works have also placed their focus on conceptual schema evolution independent of any logical data model orientation, and their investigation has concentrated on schema stability, schema quality, schema changes, and schema semantics (Gómez & Olivé, 2002, 2003). An example is the case of the work developed in Wedemeijer (2002). It proposes a framework consisting of a set of metrics that control the stability of conceptual schemas after evolution. Despite this, database evolution still remains a hot research subject.

Data model versioning is one of the solutions approved, and the versioning principles can be applied universally to many different forms of data, such as text files, database objects and relationships, and so on, despite certain negative points that lead us to consider other possible improvements. For example, the revision versioning approach generates high costs and requires much memory space as the database evolves, which will be difficult to support in the long term. Current works on database evolution prefer developing approaches that combine versioning with other approaches like the modification-versioning approach or versioning-view approach.

Concerning versioning perspectives, despite the many unresolved issues, such as refining the versioning model or speed of access to data through the different versions, versioning is gaining ground each day, and the work in schema versioning is extending. It includes both multi-temporal and spatial databases (Grandi & Mandreoli, 2003; Roddick, Grandi, Mandreoli, & Scalas, 2001).

## REFERENCES

Andany, J., Leonard, M., Palisser, C. (1991). Management of schema evolution in databases. In *Proceedings of the 17th International Conference on Very Large Databases*, San Mateo, CA. Morgan Kaufmann.

Awais, R. (2003). *Aspect-oriented database systems*. Berlin, Heidelberg, Germany: Springer-Verlag.

Connolly, T., & Begg, C. (2002). *Database systems: A practical approach to design, implementation, and management* (3rd ed.). Harlow: Addison-Wesley.

Franconi, E., Grandi, F., & Mandreoli, F. (2000, September). Schema evolution and versioning: A logical and computational characterisation. *Proceedings of the Ninth International Workshop on Foundations of Models and Languages for Data and Objects, FoMLaDO/DEMM2000*, Lecture Notes in Computer Science 2348, Dagstuhl Castle, Germany. Springer.

Gómez, C., & Olivé, A. (2002). Evolving partitions in conceptual schemas in the UML. *CaiSE 2002*, Lecture Notes in Computer Science 2348, Toronto, Canada. Springer.

Gómez, C., & Olivé, A. (2003). Evolving derived entity types in conceptual schemas in the UML. In *Ninth International Conference, OOIS 2003*, Lecture Notes in Computer Science, Geneva, Switzerland (pp. 3-45). Springer.

Grandi, F., & Mandreoli, F. (2003). A formal model for temporal schema versioning in object-oriented databases. *Data and Knowledge Engineering*, 46(2), 123-167.

Kim, W., & Chou, H.-T. (1988). Versions of schema for object-oriented databases. *Proceedings VLDB'88*, Los Angeles, CA. Morgan Kaufmann.

Lammari, N., Akoka, J., & Comyn-Wattiau, I. (2003). Supporting database evolution: Using ontologies matching. In *Ninth International Conference on Object-Oriented Information Systems*, Lecture notes in Computer Science 2817, Geneva, Switzerland. Springer.

Munch, B. P. (1995). *Versioning in a software engineering database—The change oriented way*. Unpublished doctoral thesis, Norwegian Institute of Technology (NTNU), Trondheim, Norway.

Noy, N. F., & Klein, M. (2002). *Ontology evolution: Not the same as schema evolution* (Tech. Rep. No. SMI-2002-0926). London: Springer.

Peters, R. J., & Özsu, M. T. (1997). An axiomatic model of dynamic schema evolution in object-based systems. *ACM TODS*, 22(1), 75-114.

Roddick, J. F. (1995). A survey of schema versioning issues for database systems. *Information and Software Technology*, 37(7), 383-393.

Roddick, J. F., Grandi, F., Mandreoli, F., & Scalas, M. R. (2001). Beyond schema versioning: A flexible model for spatio-temporal schema selection. *Geoinformatica*, 5(1), 33-50.

Skarra, A. H., & Zdonik, S. B. (1987). Type evolution in an object-oriented database. In B. Shriver & P. Wegner (Eds.), *Research directions in OO programming*. USA: MIT Press.

Wedemeijer, L. (2002). Defining metrics for conceptual schema evolution. In *Ninth International Workshop on Foundations of Models and Languages for Data and Objects, FoMLaDO/DEMM2000*, Lecture Notes in Computer Science 2065, Dagstuhl Castle, Germany. Springer.

## KEY TERMS

**Data Model:** Defines which information is to be stored in a database and how it is organized.

**Federated Database Schemas:** (FDBS) Collection of autonomous cooperating database systems working in either a homogenous environment, i.e., dealing with

## ***Data Model Versioning and Database Evolution***

schemas of databases having the same data model and identical DBMSs (database management systems), or a heterogeneous environment.

**Schema Evolution:** Ability of the database schema to change over time without the loss of existing information.

**Schema Integration:** Ability of the database schema to be created from the merging of several other schemas.

**Schema Modification:** Ability to make changes in the database schema. The data corresponding to the past schema are lost or recreated according to the new schema.

**Schema Versioning:** Ability of the database schema to be replaced by a new version created with one of the several existing versioning methods.

**Temporal Database:** Ability of the database to handle a number of time dimensions, such as valid time (the time line of perception of the real world), transaction time (the time the data is stored in the database), and schema time (the time that indicates the format of the data according to the active schema at that time).

**Versioning-View and Versioning-Modification:** Two combined versioning approaches that aim to create a minimum number of schema versions to reduce database access time and database space consumption.

**D**

# Data Quality Assessment

**Juliusz L. Kulikowski**

*Institute of Biocybernetics and Biomedical Engineering PAS, Poland*

## INTRODUCTION

For many years the idea that for high information processing systems effectiveness, high quality of data is not less important than the systems' technological perfection was not widely understood and accepted. The way to understanding the complexity of the data quality notion was also long, as will be shown in this paper. However, progress in modern information processing systems development is not possible without improvement of data quality assessment and control methods. Data quality is closely connected both with *data form* and *value of information* carried by the data. High-quality data can be understood as data having an appropriate form and containing valuable information. Therefore, at least two aspects of data are reflected in this notion: (1) technical facility of data processing and (2) usefulness of information supplied by the data in education, science, decision making, etc.

## BACKGROUND

In the early years of information theory development a difference between the *quantity* and the *value* of information was noticed; however, originally little attention was paid to the information value problem. Hartley (1928), interpreting information value as its psychological aspect, stated that it is desirable to eliminate any additional psychological factors and to establish an information measure based on purely physical terms only. Shannon and Weaver (1949) created a mathematical communication theory based on statistical concepts, fully neglecting the information value aspects. Brillouin (1956) tried to establish a relationship between the quantity and the value of information, stating that for an information user, the relative information value is smaller than or equal to the absolute information (i.e., to its quantity, chap. 20.6). Bongard (1960) and Kharkevitch (1960) have proposed to combine the information value concept with the one of a statistical decision risk. This concept has also been developed by Stratonovitch (1975, chaps. 9-10). This approach leads to an economic point of view on information value as profits earned due to information using (Beynon-Davies, 1998, chap. 34.5). Such an approach to information value assessment is limited to the cases in

which economic profits can be quantitatively evaluated. In physical and technical measurements, data accuracy (described by a mean square error or by a confidence interval length) is used as the main data quality descriptor (Piotrowski, 1992). In medical diagnosis, data actuality, relevance and credibility play a relatively higher role than data accuracy (Wulff, 1981, chap. 2). This indicates that, in general, no universal set of data quality descriptors exists; they rather should be chosen according to the application area specificity. In the last years data quality became one of the main problems posed by World Wide Web (WWW) development (Baeza-Yates & Ribeiro-Neto, 1999, chap. 13.2). The focus in the domain of finding information in the WWW increasingly shifts from merely locating relevant information to differentiating high-quality from low-quality information (Oberweis & Perc, 2000, pp. 14-15). In the recommendations for databases of the Committee for Data in Science and Technology (CODATA), several different quality types of data are distinguished: (1) primary (rough) data, whose quality is subjected to individually or locally accepted rules or constraints, (2) qualified data, broadly accessible and satisfying national or international (ISO) standards in the given application domain, and (3) recommended data, the highest quality broadly accessible data (like physical fundamental constants) that have passed a set of special data quality tests (Mohr & Taylor, 2000, p. 137).

## BASIC PROBLEMS OF DATA QUALITY ASSESSMENT

Taking into account the difficulty of data value representation by a single numerical parameter Kulikowski (1969) proposed to characterize it by a vector whose components describe various easily evaluated data quality aspects (pp. 89-105). For a multi-aspect data quality evaluation, quality factors like data actuality, relevance, credibility, accuracy, operability, etc. were proposed. In the ensuing years the list of proposed data quality factors by other authors has been extended up to almost 200 (Pipino, Lee, & Wang, 2002; Shanks & Darke, 1998; Wang & Storey, 1995; Wang Strong, & Firth, 1996). However, in order to make data quality assessment possible it is not enough to define a large number of sophisticated data quality factors. It is also necessary to establish the meth-

ods of multi-aspect data quality comparison. For example, there may arise a problem of practical importance—What kind of data in a given situation should be preferred: the more fresh but less credible, or outdated but very accurate ones. A comparison of multi-aspect data qualities is possible when the following formal conditions are fulfilled: (1) The quality factors are represented by non-negative real numbers, (2) if taken together, they form a vector in a linear semi-ordered vector space, and (3) they are defined so that the multi-aspect quality vector is a nondecreasing function of its components (quality factors; Kulikowski, 1986). For example, if  $t_0$  denotes the time of data creation and  $t$  is a current time, then for  $t_0 \leq t$  neither the difference  $t_0 - t$  (as not satisfying the first condition) nor  $t - t_0$  (as not satisfying the third condition) can be used as data actuality measures. However, if  $T$ , such that  $0 \leq t_0 \leq T$ , denotes the maximum data validity time then for  $t \leq T$  a difference  $\theta = T - t$  or a normalized difference  $\Theta = \theta / T$ , both satisfying the above-mentioned conditions, can be used as actuality measures characterizing one of data quality aspects. For similar reasons, if  $\Delta$  is a finite length of admissible data values and  $\delta$  is the length of data confidence interval, where  $0 < \delta < \Delta$ , then  $c = 1 / \delta$  or  $C = \Delta / \delta$ , rather than  $\delta$ , can be used as data accuracy measures.

### Comparison of Multi-Aspect Data Qualities in a Linear Vector Space

For any pair of vectors  $Q', Q''$  describing multi-aspect qualities of some single data they can be compared according to the semi-ordering rules in the given linear vector space (Akilov & Kukateladze, 1978). Linearity of the vector space admits the operations of vectors adding, multiplying by real numbers and taking the differences of vectors. For comparison of vectors a so-called *positive cone*  $K^+$  in the vector space should be established, as shown in Figure 1. It is assumed that  $Q'$  has lower value than  $Q''$  ( $Q' \prec Q''$ ) if their difference  $\Delta Q = Q'' - Q'$  is a vector belonging to  $K^+$ . If neither  $Q' \prec Q''$  nor  $Q'' \prec Q'$  then the vectors  $Q', Q''$  are called *mutually incomparable*, meaning that no quality vector in this pair with respect to the other one can be preferred. Figure 1a shows a situation of  $Q' \prec Q''$ , while Figure 1b shows vectors' incomparability caused by a narrower positive cone (between the dotted lines) being assumed.

The example shows that the criteria of multi-aspect data quality comparability can be changed according to the user's requirements: They become more rigid when the positive cone  $K^+$  established is narrower. In particular, if  $K^+$  becomes as narrow as being close to a positive half-axis, then  $Q' \prec Q''$  if all components of  $Q'$  are proportional to the corresponding ones of  $Q''$  with the coefficient of proportionality  $< 1$ . On the other hand, if  $K^+$  becomes as

large as the positive sector of the vector space, then  $Q' \prec Q''$  means that all components of  $Q'$  are lower than the corresponding ones of  $Q''$ ; in practice the domination of the components of  $Q''$  over those of  $Q'$  may be not significant, excepting one or several selected components. Between the above-mentioned two extremes there is a large variety of data quality vector semi-ordering, including those used in multi-aspect optimization theory. In particular, if  $Q' = [q_1', q_2', \dots, q_k']$  and  $Q'' = [q_1'', q_2'', \dots, q_k'']$  are two vectors, then it can be defined their scalar product:

$$(Q', Q'') = q_1' \times q_1'' + q_2' \times q_2'' + \dots + q_k' \times q_k''$$

and the length (a norm) of a vector  $Q$  given by the expression:

$$\|Q\| = \sqrt{(Q, Q)}$$

Then any fixed positive-components vector  $C$  of a norm  $\|C\| = 1$  indicates the direction of the  $K^+$  cone axis. For any given data quality vector  $Q$ , an angle  $\angle(C, Q)$  between the vectors  $C$  and  $Q$  can be calculated from its cosine given by:

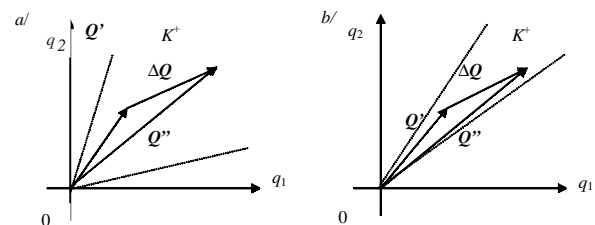
$$\cos \angle(C, Q) = (C, Q) / \|Q\|$$

This expression also can be used for data quality comparison: For the given pair of quality vectors it is assumed that  $Q' \prec Q''$  if and only if  $\cos \angle(C, Q') < \cos \angle(C, Q'')$ ; i.e., when  $Q''$  is closer to  $C$  than  $Q'$ . This type of vector comparison leads, in fact, to a comparison of weighted linear combinations of data quality factors.

### Remarks on Composite Data Quality Assessment

The above-described single data quality assessment principles should be extended to composite data and higher-order data structures. In general, it is not a trivial problem, the higher-order data structures usually being composed of various types of simple data. The quality of a record

Figure 1. Principles of comparison of vectors describing multi-aspect data values





consisting of a linearly ordered set of single data (say, a series of parameters describing the properties of a given physical object) can be defined as a nondecreasing vector function of the component data qualities (Kulikowski, 2002, p. 120). The function can be chosen in various ways. For example, if  $Q'$  and  $Q''$  are multi-aspect qualities of two data and  $q_1', q_1''$  is a pair of their first quality factors,  $q_2', q_2''$  is a pair of their second quality factors, etc., then the quality of the pair  $[Q', Q'']$  can be defined as:

$$Q = [\min(q_1', q_1''), \min(q_2', q_2''), \dots, \min(q_k', q_k'')]$$

Using other, less rigorous functions, satisfying the general conditions of multi-aspect data quality assessment is also possible. Additional vector components describing composite data quality factors also can be introduced. In particular, they may characterize composite data completeness, irredundancy and/or operability. Similar additional quality factors can be used in higher-order data structures assessment. In distributed databases, increasing attention to data interoperability as an important quality factor is paid (de La Beaujardiere, 2002; Robbins, 2002). Sugawara (2002) states that it may be a solution to provide an integrated view of heterogeneous data sources distributed in many disciplines and also in distant places. For this purpose, standardization of scientific terminology, data standards and metadata standards for document type definition are developed (Lagoze, 2002; Sugawara). Limited quality of data, in general, is not an obstacle in using them for decision making because in recent years various methods of decision making under uncertainty have been developed (Bubnicki, 2002; Kandel & Last & Bunke, 2001). However, as some authors remark (Orr, 1998; Redman, 1998; Ballou & Tayi, 1999), in well-organized databases data quality should be carefully controlled.

## FUTURE TRENDS

It can be expected that in the coming years, further progress in data quality assessment and control methods will be achieved. New methods of automatic data quality control based on sophisticated mathematical and/or logical tools and on an artificial intelligence approach will be applied. Some standard algorithms of this type will be included into commercial database management systems. High data quality requirements will be also regulated by international and national data processing standards. A basic role in this domain will be played by worldwide organizations: International Standard Organization (ISO), Committee for Data in Science and Technology (CODATA), International Federation of Information Processing (IFIP), etc.

## CONCLUSION

In most applications, high data quality is not less important than easy access to data. Data quality is, in general, independent on data volume and closely related to the data users' needs. Despite the fact that no absolute and universal data quality measure exists, a variety of approaches to data quality definition by many authors has been proposed. However, the proposals of data quality assessment are useful only if the quality factors are measurable, the vectors of quality factors are comparable, and single data quality assessment methods can be extended on higher-order data structures. For this purpose, nontrivial mathematical tools of data quality assessment should be used. They make possible choosing a data quality method that is the most suitable one to any given application domain and to the data users' requirements.

## REFERENCES

- Akilov, G. P., & Kukateladze S. S. (1978). *Ordered vector spaces* (in Russian). Nauka, Novosibirsk.
- Baeza-Yates, R., & Ribeiro-Neto B. (1999). *Modern information retrieval*. Harlow, UK: Addison-Wesley.
- Ballou, D. P., & Tayi G. K. (1999). Enhancing data quality in data warehouse environments. *Communications of the ACM*, 42(1), 73-80.
- Beynon-Davies, P. (1998). *Information systems development: An introduction to information systems engineering*. Basingstoke: Macmillan Press Ltd.
- Bongard, M. M. (1960). On the concept of "useful information" (in Russian). *Problemy kibernetiki*, vol. 4, Moscow.
- Brillouin, L. (1956). *Science and information theory*. New York: Academic Press.
- Bubnicki, Z. (2002). *Uncertain logics, variables and systems*. Berlin, Germany: Springer.
- de La Beaujardiere, J. (2002). Interoperability in geospatial Web services. *18<sup>th</sup> International CODATA Conference: Book of Abstracts*, Montreal, Canada (p. 8).
- Hartley, R. V. L. (1928). Transmission of information. *Bell System Technical Journal*, 7(3), 535-563.
- Kandel, A., Last, M., & Bunke, H. (2001). *Data mining and computational intelligence*. Heidelberg, Germany: Physica-Verlag.
- Kharkevitch, A. A. (1960). On information value (in Russian). *Problemy kibernetiki*, vol. 4, Moscow.

Kulikowski, J. L. (1969). *Statistical problems of information flow in large-scale control systems*. IV Congress of the IFAC, Technical Session 63, Warsaw.

Kulikowski, J. L. (1986). Principles of data usefulness evaluation (in Polish). In *Technology and Methods of Distributed Data Processing, Part I* (pp. 35-51). Vroclav: Vroclav Technological University,.

Kulikowski, J. L. (2002). Multi-aspect evaluation of data quality in scientific databases. *18th International CODATA Conference: Book of Abstracts*, Montreal, Canada (p. 120).

Lagoze, C. (2002). The Open Archives Initiative: A low-barrier framework for interoperability. *18th International CODATA Conference: Book of Abstracts*, Montreal, Canada, 9.

Mohr, P. J., & Taylor, B. N. (2000). Data for the values of the CODATA fundamental constants. *17th International CODATA Conference: Book of Abstracts*, Baveno, Italy, 137.

Oberweis, A., & Perc, P. (2000). Information quality in the World Wide Web. *17th International CODATA Conference: Book of Abstracts*, Baveno, Italy, 16.

Orr, K. (1998). Data quality and systems theory. *Communications of the ACM*, 41(2), 66-71.

Piotrowski, J. (1992). *Theory of physical and technical measurement*. Amsterdam: Elsevier.

Pipino, L. L., Lee, Y. W., & Wang, R. Y. (2002). Data quality assessment. *Communications of the ACM*, 45(4), 211-218.

Redman, T. C. (1998). The impact of poor data quality on the typical enterprise. *Communications of the ACM*, 41(2), 79-82.

Robbins, R. J. (2002). Integrating bioinformatics data into science: From molecules to biodiversity. *18th International CODATA Conference: Book of Abstracts*, Montreal, Canada, 1.

Shanks, G., & Darke, P. (1998). A framework for understanding data quality. *Journal of Data Warehousing*, 3(3), 46-51.

Shannon, C. E., & Weaver, W. (1949). *The mathematical theory of communication*. Urbana: University of Illinois Press.

Stratonovitsch, R. L. (1975). *Teoria informacii* (in Russian). Sovetskoe Radio, Moscow.

Sugawara, H. (2002). Interoperability of biological data resources. *18th International CODATA Conference: Book of Abstracts*, Montreal, Canada, 9.

Wang, R. Y., Storey, V. C., & Firth, C. P. (1995). A framework for analysis of data quality research. *IEEE Transactions on Knowledge and Data Engineering*, 7(4), 623-641.

Wang, R. Y., & Strong, D. (1996). Beyond accuracy: What data quality means to data consumers. *Journal of Management Information Systems*, 12(4), 5-34.

Wulff, H. R. (1981). *Rational klinik*. Munksgaard, Copenhagen.

## KEY TERMS

(The symbol  $\rightarrow$  refers to the definitions of other terms defined below.)

**Composite Data:** Data containing an ordered string of fields describing several attributes (parameters, properties, etc.) of an object.

**Data Accuracy:** An aspect of numerical ( $\rightarrow$ ) data quality: a standard statistical error between a real parameter value and the corresponding value given by the data. Data accuracy is inversely proportional to this error.

**Data Actuality:** An aspect of ( $\rightarrow$ ) data quality consisting in its steadiness despite the natural process of data obsolescence increasing in time.

**Data Completeness:** Containing by a composite data all components necessary to full description of the states of a considered object or process.

**Data Content:** The attribute(-s) (state[-s], property[-ies], etc.) of a real or of an assumed abstract world to which the given data record is referred.

**Data Credibility:** An aspect of ( $\rightarrow$ ) data quality: a level of certitude that the ( $\rightarrow$ ) data content corresponds to a real object or has been obtained using a proper acquisition method.

**Data Irredundancy:** The lack of data volume that by data recoding could be removed without information loss.

**Data Legibility:** An aspect of ( $\rightarrow$ ) data quality: a level of data content ability to be interpreted correctly due to the known and well-defined attributes, units, abbreviations, codes, formal terms, etc. used in the data record's expression.

**Data Operability:** An aspect of (→) data quality: a level of data record ability to be used directly, without additional processing (restructuring, conversion, etc.).

**Data Precision:** An aspect of numerical (→) data quality: the maximum error between a real parameter and its value given by the data, caused by the data values' discretization. Data precision is inversely proportional to this error.

**Data Quality:** A set of data properties (features, parameters, etc.) describing their ability to satisfy the user's expectations or requirements concerning data using for

information acquiring in a given area of interest, learning, decision making, etc.

**Data Relevance:** An aspect of (→) data quality: a level of consistency between the (→) data content and the area of interest of the user.

**Data Structure:** Formal description of a (→) composite data indicating the order, contents, lengths and lists of attributes' values of its fields.

**Simple Data:** Data consisting of its identifier and a field describing a single attribute (parameter, property, etc.) of an object.

# Data Warehouses

**Antonio Badia**

*University of Louisville, USA*

## INTRODUCTION

Data warehouses (DW) appeared first in industry in the mid 1980s. When their impact on businesses and database practices became clear, a flurry of research took place in academia in the late 1980s and 1990s. However, the concept of DW still remains rooted on its practical origins. This entry describes the basic concepts behind a DW while keeping the discussion at an intuitive level. The entry is meant as an overview to complement more focused and detailed entries, and it assumes only familiarity with the relational data model and relational databases.

## BACKGROUND

Databases in the 1970s and 1980s were used mainly to maintain data for everyday operations. A typical example is a bank database holding information about accounts that is used by a network of ATM machines. This is called *operational data*, and the role of the database is mainly to support *transactions*. Transactions are operations that access and change the data in the database; in the example of the bank, an ATM may access the database to check if a client has enough money in her/his account, and to change the balance if a withdrawal or deposit takes place. While transactions usually affect only small parts of the database, databases have to handle efficiently a large number of transactions, many times concurrently. Thus, the database architecture is geared towards efficient support of small, localized access. This is called *on-line transaction processing*, or OLTP. However, in the mid 1980s, emphasis shifted towards comprehensive analysis of current and historical data, in order to understand business patterns. This analysis is high level and involves many related low-level items and has as its goal to support decision making. Hence, this kind of analysis is called *decision support* (DS), but the term *on-line analytical processing*, or OLAP, is also used, to emphasize the differences with OLTP (Kimball & Strehlo, 1995). The typical DS task involves summarizing large amounts of low-level data and relating different aspects of the business to find interesting correlations, and therefore database access is based on complex queries. Because

of the business intelligence it provides, OLAP grew rapidly and nowadays is a necessary tool for most medium and large size enterprises.

However, the analysis that DS relies upon is made complicated by a historical factor. In many organizations, especially those of a certain size and complexity, data was stored in several separated data sources, because each unit within the organization developed its own database to support its informational needs. Decision support requires that all the data in the different databases is put together to yield a global picture of the organization. As an example, imagine a hospital where the surgery unit has developed a database of patients undergoing surgery (with information on the type of surgery), Pharmacy has developed a database of patients that are administered some medication (with information on the dosage, time, etc.), and accounting has developed a database of patients in order to bill them. If the hospital's management asks for an analysis that correlated patient's socioeconomic status with type of surgery and amount of medication prescribed, the information needed is dispersed throughout all three databases. If patients are identified by social security number in one database, patient ID in another, and a combination of name and date of birth in another, putting the information together may be complicated. Thus, many companies decided to centralize and consolidate all their data in one central repository: a DW. In the following, we describe the main characteristics of a DW, its design and implementation.

## DATA WAREHOUSING

### Data Warehouse Design: The ETL Process

A DW contains a copy of the data from other databases, which are called the *data sources* of the warehouse. In order to get the data from those data sources (usually OLTP databases) and allow them to continue working normally, the data from the sources is copied inside the DW at regular, preestablished intervals, to refresh the DW. Because there are multiple data sources, it is necessary to watch out for redundant (duplicate) data,

missing data, or heterogeneous data. This is done during the *extraction, Translation, and loading* (ETL) process (Immon, 2002; Kimball & Ross, 2002).

The ETL process involves extraction, transformation, integration, cleansing, loading and computation of additional data (different authors give somewhat different phases, or name them differently). Extraction refers to the act of capturing the data from the sources and physically sending it to the DW. Usually, standard external interfaces supported by the databases (called *gateways*) are used. To make the process efficient, it is desirable to keep track of which data is copied in the DW on a given extraction, so that only new data is copied on the next extraction.

Different databases (especially historical ones, or legacy systems) may contain data which is represented in different formats. Thus the *transformation* step, which involves converting data to a uniform format, removing, adding and reordering attributes if necessary (e.g., adding a key, or a timestamp).

Because different databases were created by different people for different purposes, it is likely that they contain related (or overlapping) information stored in different ways. When the data is integrated, there may be problems of *homonyms* (i.e., the same name for different concepts), *synonyms* (i.e., two names for the same concept), and many other semantic mismatches. Thus, another step in extraction is to *integrate* the data, that is, to merge and match data to ensure that data about the same entity is integrated and data about different entities is separated (our previous hospital example provides an instance of such semantic mismatches). A separate step is *data cleansing*, which involves making sure that the data is as devoid of noise as possible; typos, data entry errors, missing data are dealt with at this stage.

Finally, the data must be loaded into the DW. This is a complicated task because, in normal operating mode, the DW is queried and the data is not changed to maximize performance answering queries. Thus, the data in the DW is considered static and left to be out of sync for the time between refreshes. Then, a large amount of data must be loaded in the DW at once to refresh it (batch load); this is called *DW refreshing*. The window of opportunity depends on DW usage. At loading time, other activities that will help improve query processing must be carried out (e.g., sorting, summarizing, indexing, partitioning, checking integrity constraints). As a result, loading the DW is a complex process in itself. Because of large volumes and shrinking time windows, most commercial utilities use *incremental* approaches, in which old results are reused and only truly new data is added (Jarke, Lenzerini, Vassiliou, & Vassiliadis, 2000).

To manage the DW, a *metadata repository* is created. This is a system catalog that contains metadata

(i.e., information about the data in the DW, including their origin, format, intended meaning, range, and information about the source of the data). The catalog is usually large and complex, because it used to support all steps in the ETL process. Thus, it is not surprising that sometimes the metadata repository is kept in a database itself, and that building it is one of the most difficult and important steps in the process of building a DW.

Currently, ETL is often done off-line, by hand, and by replicating almost everything. There are tools to help with this task: data migration tools allow simple transformation rules to be applied to the data (“replace string *gender* with string *sex*”); *data scrubbing* tools use domain-specific knowledge to do the cleansing (“ages should be between 18 and 65”). They often use parsing and fuzzy matching techniques to add some intelligence to an otherwise complex and laborious process. The tools that different systems offer to handle ETL vary enormously from system to system (IBM, 2004; Oracle, 2004).

## Data Warehouse Design: Logical and Physical Models

The basic DW design is influenced not by theories of *normalization*, as in regular databases, but by the fact that the DW goal is to provide information about a business model.

The data in a DW is analyzed in terms of *facts* and *dimensions*. A *fact* is a basic business datum, contains raw data, and is irreducible (e.g., point-of-sale information: who bought what, when, where and at what price is the basic fact on a retail enterprise). A *dimension* is an attribute of the fact (e.g., Client, Product, Time, and Store are dimensions of the point-of-sale fact). Usually, there are several dimensions to every fact. Each dimension can have in turn a set of associated attributes. For instance, Store may have attributes Name, City, and State associated with it. It is also typical that the values of a dimension can be organized in a hierarchy. The typical example is the dimension *Time*, which can be organized in Date, Week, Month, Quarter and Year. The previous example, the dimension Store, can also be organized in a geographical hierarchy through City and State. Finally, besides dimensions, a fact contains *measures*, usually numerical values indicating some properties of the fact. For instance, in our point-of-sale example, Price, Quantity and Amount may be three dimensions. Obviously, these definitions are informal; what is fact and what is a dimension depends on the DW subject and on the business analysis that the DW is to support (hence the usual statement that DW design is subject oriented).

The basic data model choices are relational OLAP (ROLAP) and multidimensional OLAP (MOLAP).



ROLAP uses relational technology to support the DW design. In ROLAP, information about the dimension values is maintained in the *dimension tables*. Usually, there is one dimension table per dimension. In our example, Store, Client, and Product may have one table each. Information about facts is organized in a table, called the *fact table*. Each row contains one fact, which is represented by references to the dimensions (technically, one foreign key for each dimension table) and by measures. Thus, rows in a fact table tend to be “narrow” and contain a mix of foreign keys and raw numeric data. This table is typically very large since there are many particular facts to be stored. In the example, a large chain of stores will have millions of point-of-sale transactions over a period of several months; the longer the period, the more sales facts. Also, the fact table tends to grow quickly (there will be many daily sales that must be added to this table). On the other hand, dimension tables tend to be much smaller (orders of magnitude) than the fact table, and tend to change at a much slower rate. Clearly, there is going to be one row per store in the table Store, and that is not going to add up to millions of rows, as in the fact table. Also, a new row will not be added unless a new store is opened.

When all the information about a dimension is stored in one table, the resulting schema is called a *star schema* (picture the large fact table in the middle and the smaller dimension tables around it). In some cases, this design leads to a high level of redundancy because dimension tables may be *denormalized* (i.e., not in a normal form as they would be in the usual relational database design; Date, 2000). The dimension tables can be normalized by splitting the information in them in several tables; the resulting design is called a *snowflake* schema (as before, picture a large fact table in the middle surrounded by dimension tables. But now, each dimension table in turn may be surrounded by a number of smaller tables). Because the fact table is large to start with, it is usually normalized, to avoid redundancy that would increase its size even more. Technically, each dimension table has a primary key, which is also included in the fact table as a foreign key. The combination of all foreign keys (one per dimension) becomes the primary key of the fact table.

The dimension table stores hierarchies by using the elements of the hierarchy as attributes; for instance, the table *Time* may have attributes Date, Week, Month, Quarter, and Year in every row. A combination of such values tells us where in the hierarchy we are.

Besides these tables, the DW may contain *summarized* or *aggregate* tables. These are tables that contain summarized information (i.e., facts that have been grouped by some of the dimensions). For instance, we may have a table that summarizes sales by store, by city, or by state.

In MOLAP, facts with  $n$  dimensions are organized in an  $n$ -dimensional cube. For instance, the dimensions of a sale measure may be store, product and time. Thus, the information can be organized in a three-dimensional cube. All the dimensions together are assumed to uniquely determine the measure (a particular store, product and time give us a unique sale). Thus, store  $i$ , product  $j$  and time  $l$  give a cell  $[i][j][k]$  that has as content the measure(s) for that sale. The measures themselves are usually numerical (e.g., amount of sale). Clearly, each cell in the cube corresponds to a row in a fact table. As before, dimensions are described by a set of attributes (store may be described by store ID, manager, city, state); and some attributes may be organized in hierarchies (city and state form a geographical hierarchy). The cube is stored as an  $n$ -dimensional array in the computer. This representation has a great problem with *sparseness*, the fact that most combinations of dimensions do not have an associated measure (i.e., not all products are sold at all stores at all times); several compression schemes are used in this model to avoid having to store large, mostly empty arrays. This model is influenced by the success of spreadsheet programs in business analysis. However, most DWs nowadays use the ROLAP approach, since it allows leveraging all the know-how and software already existing in relational database systems.

The DW design process is a complex one. Different authors use different labels and recommend different steps to accomplish DW design (Immon, 2002; Kimball & Ross, 2002); we show here a generic framework that encompasses all necessary steps:

1. **Define a common business model:** agree on subjects (clients, products, etc.), their attributes and properties (time granularity, needed information, etc.), relationships among them, definition of basic terms (Can a client be also a provider? What is a month: 4 weeks or a calendar month?). Every stakeholder has to agree on this model, which makes this step a vast and complex effort.
2. **Logical design:** Choose central facts and dimensions. For each fact, identify measures. For each dimension, identify attributes and granularity.
3. **ETL design:** Find source(s) for each datum (e.g. attributes, measures). Design and implement the ETL process. Design and populate the metadata repository.
4. **Physical design:** Choose auxiliary data to store (e.g., aggregates to precompute, indices needed, data placement, partitioning).

5. Define front-ends for different users, giving them direct (SQL) access or other interfaces.
6. **Optional:** Design one or more data marts. A data mart is a database containing a subset of the DW. Usually created with a strong business focus, it yields a database smaller (and more manageable) than the DW for a group of users that have concrete, narrow, business needs. Note that the data mart is dependent on the DW for its contents, just like the DW is dependent on several data sources. Data marts are usually refreshed whenever the DW is refreshed; therefore, the ETL process has to take into account any existing data marts.

## Querying the Data Warehouse

The key idea of DW is to collect data in advance of queries. Thus, typically DWs contain consolidated data from many sources. Also, the data in the DW usually spans long periods of time (historic data is needed to analyze trends over time). Moreover, the data is time sensitive, since the DW keeps history of many data (sets of snapshots), which are updated periodically. Hence, most data in a DW is *time-stamped*. Also, DWs tend to keep raw (low-level) data because different users require different levels of analysis and information granularity. Clearly, low-level data can be summarized to produce high-level analysis, but a high-level summary cannot produce low-level data.

All these factors together explain why DWs are one or several orders of magnitude larger than traditional OLTP databases. Although OLTP databases exist in the megabyte to (a few) gigabyte range, DWs usually live in the terabyte range. Because queries to the DW are posed by upper management, which needs the answer to take decisions in a timely manner, and the queries involve complex analysis of large amounts of data, answering those queries very quickly is both a necessity and a challenge. Hence, a great deal of the implementation of a DW is geared towards this task. For lack of space, we will not discuss here implementation techniques; we will simply point out that, special performance techniques, beyond what is usually available in a database management system, are necessary in a DW environment (Sasha & Bonnet, 2002).

Normally, one queries the DW by asking for values of measures for some specific values of dimensions, for instance sales by store by month or sales of clothing articles by state. Some of the more common operations of DW follow:

- **Aggregating:** A measure over one or more dimensions, that is, finding an aggregate over a value of a dimension (e.g., total sales per week, state).

When we aggregate over the hierarchy of a dimension the operation is called *roll-up*. For instance, given the Store dimension, and the total sales per store, asking for the total sales per state is a roll-up. The inverse is *drill-down*: Given summarized information, ask for more detail. Thus, given total sales by state, asking for total sales by store is a drill-down.

- **Pivoting:** That is, taking a tabular representation of the data and turning it into an  $n$ -dimensional chart (spreadsheet) where the labels in the axes are the values of the original table. For instance, the fact table Sales could be pivoted by Time and Store, to give a two-dimensional table, with Time in one axis and Store on the other, and the information for sales on the cells (i.e., each cell contains the sales of a certain store on a certain period). As in a spreadsheet, we would have partial and total sums. The result of pivoting is called *cross-tabulation*. Pivoting can be combined with other operations.
- **Slicing and Dicing:** Slicing is to do a selection (with an equality condition) on one or more dimensions and, dicing is to do a selection (with a range condition) on one or more dimensions. The terminology comes from the MOLAP model, since selecting a single value of a dimension is like taking a slice of the cube, and selecting a range generates a smaller cube, part of the original one.
- **Time Analysis:** The time dimension is especially important in DW. Because trends develop over time, time series, progressions, histories, and so forth are very common and important types of analysis.

The type of queries used in DS environments has highlighted weaknesses of SQL. Therefore, several SQL extensions have been proposed. We highlight some important extensions that have become part of the latest (SQL3) standard (Melton & Simon, 2000).

- **Extended aggregate functions:** Rank, percentile, and cumulative total median are among the new aggregate functions.
- **Cubes:** Because each roll-up corresponds to a single SQL query with grouping, and rolling up  $n$  dimensions can be done with any subset of them (so there are  $2^n$  possibilities) we have  $2^n$  possible SQL queries. Computing these queries is wasteful, since they have a lot in common. A proposed extension to SQL is the Cube (or Data Cube) operator, which is equivalent to a collection of GROUP BYs. A cube is a summary of data by dimensions; for instance, a cube over attributes

(Product, Year, City) would generate aggregates over (Product, Year, City), (Product, Year), (Product, City), (Year, City), (Product), (Year) and (City). The ROLL-UP operator is a specialization of this: the same cube as before would generate aggregates over (Product, Year, City), (Product, Year) and (Product) only.

- **Window operator:** Many interesting business questions arise from the comparison of several groups of items. For instance, a moving average computes averages for different but overlapping sets of numbers. In traditional SQL, the groups of numbers would be formed by using the GROUP BY operator, and could only be non-overlapping and static. The window operator allows the application of aggregates over related, dynamic groups of numbers.

## FUTURE TRENDS

There are currently some indications of where DWs will be headed in the next few years. The main ones can be summarized as follows:

- **Even larger sizes:** As the ability to collect and store more information in digital form grows (think of RFID tags, for instance), the size of warehouses will continue to grow. This means we will see more research on query optimization issues (Theodoratos, 2003), parallel architectures, logical and conceptual models that help connect the DW design to its business goals more tightly (Weir, Peng, & Kerridge, 2003) to keep performance up with size.
- **Incorporation of multimedia data:** Just like regular databases have started recently to incorporate data beyond alphanumeric format (e.g., audio, video, free text), so is the push for DWs to deal with these same sources of information.
- **Connection with the Web:** As Web mining is seen as an opportunity to gather information for competitive advantage by business organizations, the warehouses will be used to contain and analyze the enormous amount of information created by e-commerce and other Web-based forms of commerce. New issues that are already being investigated include extending the scope of DW to deal with XML data (Facca & Lanzi, 2003; Vrdoljak, Banek, & Rizzi, 2003).
- **Connection with data mining:** Already the favourite application (because the DW gathers enough information to allow for discovery of pat-

terns and trends), tighter integration is already a goal of current research (Tsur et al., 1998; Wang, Yang, & Yu, 2002).

## CONCLUSION

Data warehouses are repositories of information that maintain a centralized copy of the contents of several enterprise databases. They make a global, in-depth analysis of the enterprise possible and therefore are of great value to business intelligence. However, DWs are difficult to design and implement. In a DW environment, it is necessary to decide in advance what to store; the integration problems may be very difficult to overcome, and the architecture does not respond well to changes. Due to its strategic value, DWs will continue to be an asset to medium and large size companies in the foreseeable future. Research in DW certainly continues to attract strong interest, with dedicated conferences like DAWAK (Kambayashi, Mohania, & Wob, 2003), DOLAP (DOLAP, 2003) and DMDW (Lenz, Vassiliadis, Jeusfeld, & Staudt, 2003). As stated in the previous section, new areas of research will expand DW scope and will make them, if anything, even more important.

## REFERENCES

- Bouguettaya, A., Benatallah, B., & Elmagarmid, A. (1998). *Interconnecting heterogeneous information systems*. Reading, MA: Kluwer Academic Publishers.
- Date, C. (2000). *An introduction to database systems* (7th Ed.). Reading, MA: Addison-Wesley.
- DOLAP. (2003). *ACM 6th International Workshop on Data Warehousing and OLAP (DOLAP)*. New Orleans, LA: ACM Press
- Facca, F. M., & Lanzi, P. L. (2003). *Recent developments in Web usage mining research*. DAWAK.
- Gray, J., Bosworth, A., Layman, A., & Pirahesh, H. (1996). DataCube: A relational aggregation operator generalizing group by, cross-tab, and sub-totals. In *Proceedings of the 12th IEEE ICDE*. New Orleans, LA: IEEE Computer Society.
- IBM. (2004, June). Data warehousing center administration guide [Computer software manual]. Retrieved from <http://www-306.ibm.com/software/data/db2/datawarehouse/>
- Immon, W. H. (2002). *Building the data warehouse* (3<sup>rd</sup> Ed.). New York: Wiley & Sons.

Jarke, M., Lenzerini, M., Vassiliou Y., & Vassiliadis, P. (2000). *Fundamentals of data warehouses*. Springer-Verlag.

Kambayashi, Y., Mohania, M., & Wob, W. (Eds.). (2003). Data warehousing and knowledge discovery. In *Proceedings of the 5th International Conference in Data Warehousing and Knowledge Discovery (DAWAK)*, Prague, Czech Republic.

Kimball, R., & Ross, M. (2002). *The data warehouse toolkit* (2<sup>nd</sup> Ed.). New York: Wiley & Sons.

Kimball, R., & Strehlo, K. (1995). Why decision support fails and how to fix it. *SIGMOD Record*, 24(3), 92-97.

Lenz, H.-J., Vassiliadis, P., Jeusfeld, M., & Staudt, M. (Eds.). (2003). Design and Management of Data Warehouses. *Proceedings of the 5th International Workshop*, Berlin, Germany.

Melton, J., & Simon, A. R. (2002). *SQL 1999: Understanding relational language components*. San Francisco: Morgan Kaufmann.

Oracle. (2004, June). Oracle9i Warehouse Builder Release 9.2 [Computer software manual]. Retrieved from <http://otn.oracle.com/documentation/warehouse.html>

Sasha, D., & Bonnet, P. (2002). *Database tuning: Principles, experiments and troubleshooting techniques*. San Francisco: Morgan Kaufmann.

Theodoratos, D. (2003). *Exploiting hierarchical clustering in evaluating multidimensional aggregation queries*. DOLAP.

Tsur, S., Ullman, J. D., Abiteboul, S., Clifton, C., Motwani, R., Nestorov, S., et al. (1998). Query flocks: A generalization of association-rule mining. In *Proceedings of the ACM SIGMOD Conference*. Seattle, WA: ACM Press.

Vrdoljak, B., Banek, M., & Rizzi, S. (2003). In Kambayashi et al. (Eds.), *Designing Web warehouses from XML schemas*.

Wang, H., Wang, W., Yang, J., & Yu, P. S. (2002). Clustering by pattern similarity in large data sets. *Proceedings of the ACM SIGMOD Conference*. Madison, WI: ACM Press.

Weir, R., Peng, T., & Kerridge, J. (2003). *Best practice for implementing a data warehouse: A review for strategic alignment*. DMDW.

Wolfgang, H., Bauer, A., & Harde, G. (2003). *Xcube: XML for data warehouses*. DOLAP.

## KEY TERMS

**Dimension:** A property of a fact that specifies or explains one aspect of said fact. Usually a dimension has information associated with it.

**ETL:** Extraction-transformation-load is a process by which the DW is (re)populated from data in the data sources. During the process, relevant data is extracted from the source, adequately processed for integration and loaded into the DW.

**Fact:** Basic, irreducible data item that is stored in the DW. It represents the basic unit of business analysis and therefore must represent the basic activity of the enterprise under consideration.

**MOLAP:** Multidimensional OLAP is a method of implementing a DW that relies on a multidimensional view of data. Information is seen as a cube in  $n$ -dimensions; tailored structures are used to store it, and tailored query languages are used to access it.

**OLAP:** On-line analytical processing in an analysis of data characterized by complex queries that aim at uncovering important patterns and trends in the data in order to answer business questions.

**ROLAP:** Relational OLAP; method of implementing a DW that relies on relational technology. In ROLAP, the warehouse is implemented as a relational database, using tables to deposit the information and relational technology (SQL) to access it.

**Snowflake Schema:** Star schema in which the dimension tables have been normalized, possibly leading to the creation of several tables per dimension.

**Star Schema:** Relational database schema that results from ROLAP; facts and dimensions are stored in tables. The fact table is considered the core of the resulting database, connected to each and every dimension table. When visualized with the fact table in the center, and the dimension tables surrounding it, they create the figure under which the schema is named.



# Data Warehousing and OLAP

**Jose Hernandez-Orallo**

*Technical University of Valencia, Spain*

## INTRODUCTION

Information systems provide organizations with the necessary information to achieve their goals. Relevant information is gathered and stored to allow decision makers to obtain quick and elaborated reports from the data.

A data warehouse is an especially designed database that allows large amounts of historical and contextual information to be stored and accessed through complex, analytical, highly aggregated, but efficient queries. These queries capture strategic information, possibly presented in the form of reports and support management decision making. As we will see, data warehouses differ from general, transactional databases in many ways.

## BACKGROUND

The pristine motivation for building an information system was to integrate the information that would be necessary to support decision making. With the advent of databases and transactional applications, the main goal of information systems drifted towards the organization of data in such a way that software applications work. This drift triggered a specialization of database technology to serve transactional information systems, evolving database design and implementation, as well as database management systems (DBMSs), towards this end. The kind of work performed on these databases was called on-line transactional processing (OLTP).

However, the need for decision-making support did not fade away. On the contrary, more and more organizations were requiring further analytical tools to support decision making: reporting tools, EIS (executive information systems), and other DSS (decision support systems) tools, which many DBMS vendors began to call “business intelligence.” These tools were able to exploit the database in a more powerful way than traditional query tools. These new tools made it easier to aggregate information from many different tables, to summarize data in a much more powerful way, and to construct textual and graphical reports over them, full of condensed, graphical, and statistical information. This other kind of work that was performed on the database was called on-line analytical processing (OLAP).

Although the previous scenarios, working on the same physical database, can still be found in many organizations all over the world, there are three important problems associated with this approach.

- First, and most important, is the use of the same database for both OLTP and OLAP. OLAP queries are usually complex, considering much historical data and interweaving many tables with several levels of aggregations. Many OLAP queries are “killer” queries, requiring many resources, and may highly disturb or even collapse the transactional work. Because of this, many reports and complex queries were run at night or during the weekends.
- Second, the data stored in the transactional database are the data required only by the applications. Data that are not to be used by any application are not stored. Additionally, many transactional databases remove or simply do not include historical data. On the other hand, adding all the historical data and other sources of information for OLAP work in the same transactional database can be an important overhead storage for the transactional work.
- Third, the design and organization of the database is specialized for OLTP: Normalization is common, indexes are created for improving transactions, and so forth. However, these choices might not be good for OLAP operations. This means that even with a separate, devoted database (i.e., a replica), OLAP will not be efficient for large databases.

The previous problems stimulate the construction of separate data repositories, specialized for analytical purposes. In the early nineties, these repositories were called data warehouses and its associated technology data warehousing (Inmon, 1992). The attention widened from enterprises and vendors, but it was not until the late 1990s that the academic world paid attention. All this and the appearance of more mature tools turned data warehousing into a new database discipline on its own.



## DATA WAREHOUSES

The establishment of data warehousing as a new discipline has much to do with the fact that, once one database (the transactional) has been clearly separated from the other (the historical/analytical), they are quite different kind of “beasts.” Table 1 shows the strong differences between them. The special characteristics of the data and operations in the data warehouse case has led to a specialization in the data warehousing technology, establishing new designing paradigms, adopting other data models, operators, and implementation tools.

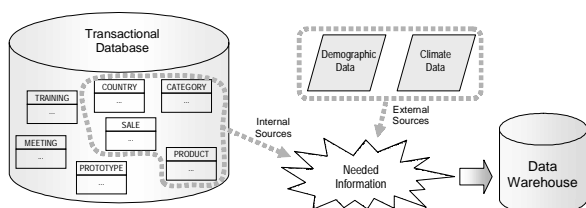
Having seen the advantages of constructing a separate data repository, the first question is to determine the data that will be included in the repository. This will surely depend on the analysis of the data requirements for the business intelligence applications that will use the data warehouse, such as the OLAP tools, the DSS systems, and, possibly, data mining. Frequently, this means that part of the information existing in the transactional databases (but not all and not at the same level of detail) has to be loaded into the data warehouse, as well as any external data that could be useful for the analytical processing. Figure 1 illustrates this integration.

Integrating data from several sources, with different formats, data models, and metadata, is not an easy task. Integration from different sources has always a negative effect on data quality; missing data are created because

Table 1. Differences between transactional databases and data warehouses

	Transactional Database	Data Warehouse
Purpose	Daily operations. Support to the software applications.	Information retrieval, reports, data analysis.
Data Characteristics	Data about the organization inner working, changing data, internal data, incomplete data.	Historical data, internal and external data, descriptive data.
Data Models	Relational, object-relational, normalized.	Multidimensional, snowflake, partially denormalized.
Users	Hundreds/thousands: applications, operators, administrator, ...	Dozens: managers, executives, analysts (farmers and explorers).
Access	SQL. Read and write.	SQL and specific OLAP operators (slice & dice, roll, pivot, ...). Read-only.

Figure 1. Data warehouses integrate data from internal and external sources



some features exist in some sources but not in others, as well as incompatible formats, units, and so forth. It is extremely important that data are transformed into unified formats and schema, and cleansed from missing and anomalous data. We will review these issues, but next we must address another crucial issue, the way in which information is organized in a data warehouse.

## Data Warehouse Organization: The Multidimensional Model

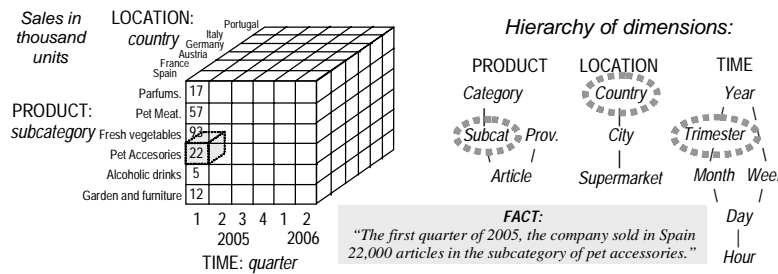
One idea for constructing a data warehouse can be to apply a traditional design, using an entity-relationship or UML-like conceptual model, and transform it into a relational or an object-relational database, integrating all the sources of information into one highly normalized, off-the-shelf database. However, this has been shown to be a bad way for organizing data in a data warehouse. The reason is that classical data models, and especially the relational model, are well suited for performing transactional work but do not deal well with complex queries with a high degree of aggregation.

As a consequence, the most widespread model for data warehouses is a different data model: the multidimensional model (Golfarelli, Maio, & Rizzi, 1998). Data under the multidimensional model are organized around *facts*, which have related *measures* and can be seen in more or less detail according to certain *dimensions*. As an example, consider a multinational European supermarket chain. Its basic facts are the sales, which can have several associated measures: total income amount, quantity, number of customers, and so forth, and can be detailed in many dimensions: the time of purchase, the product sold, the place of the sale, and so forth. It is enlightening to see that measures usually correspond to the question, how much and how many? whereas the dimensions correspond to the questions *when*, *what*, and *where*.

The interesting thing about the dimensional model is that it eases the construction of queries about facts at several levels of aggregation. For instance, the fact “the sales summed up 22,000 products in the subcategory pet accessories in all the supermarkets in Spain in the first quarter of 2005” represents a measure (quantity = 22,000) of a sale with granularity month for the time dimension (first quarter of 2005), with granularity country for the location dimension (Spain) and with granularity subcategory for the product dimension (pet accessories).

The structure of possible aggregations on each dimension constructs a hierarchy from which we can select any granularity. For instance, as shown in Figure 2, the dimension time is hierarchical, existing several

Figure 2. A fact shown in a data cube selected from a hierarchy of dimensions.



paths from the most fine-grained resolution (hour) to the most coarse-grained resolution (year). The kind of hierarchies that we may have gives several names to the whole: simple star, hierarchical star, or snowflake. The example in Figure 2 is a snowflake, because there are many levels in each dimension and there are alternative paths.

The previous *aggregations* on the dimensions does not fix the condition in each dimension, it just settles the “data cube” resolution. As we can see in Figure 2, a data cube just specifies at which level of resolution the selection can be made. In the example of Figure 2, we have a data cube with resolution country for the location dimension, subcategory for the product dimension, and quarter for the time dimension. It is in this data cube where we can make a *selection* for a specific location (Spain), product (pet accessories) and time (first quarter of 2005).

Finally, for many data warehouses it is not possible to organize all the information around one fact and several dimensions. For instance, a company might have one kind of facts and dimensions for sales, another for invoicing, another for personnel. The solution is to create similar substructures for each of these areas. Each substructure is called a data mart. A data warehouse under the multidimensional model is then a collection of data marts.

### Exploitation: OLAP Operators and Tools

A data model comprises a set of data structures and a set of operators over these structures. In the previous section we have seen that the data structures in the dimensional model are the facts with their measures, and the dimensions with their hierarchies and attributes for each level. We have seen that a single “operator” can be defined by just choosing a measure from the fact and a level for each dimension, forming a data cube, and then, selecting the values of one or more dimensions.

Query tools on data warehouses under the multidimensional model usually have a graphical interface, which

allow the user to select the data mart, to pick one or more measures for the facts (the aggregated projection), to choose the resolution in the dimension hierarchies and to express additional conditions.

As we have discussed, there can be more than three dimensions. This would prevent the graphical representation of data cubes in many situations. On the contrary, the relational model (as well as other classical models) has the advantage that the result of a query is a table, which can always be represented on a screen or a paper. The typical solution taken by the query tools for the dimensional model is a hybrid between tables and cubes. For instance, the cube of Figure 2 can be represented in two dimensions, as shown in Table 2.

The idea is that two or more dimensions can be “dragged” to the left or to the top, and we can have an arbitrary number of dimensions in a bidimensional table. This is feasible precisely because we are obtaining highly summarized (i.e., aggregated) information and the number of rows and columns is manageable. These hybrid tables are usually called “reports” in OLAP tools.

With the multidimensional model and this hybrid data cube/table representation in mind it is easier to understand some additional operators that are more and more common in OLAP tools, called OLAP operators:

- **Drill:** de-aggregates the data (more fine-grained data) following the paths of one or more dimensions.
- **Roll:** aggregates the data (more coarse-grained data) following the paths of one or more dimensions.
- **Slice & dice:** selects and projects the data into one of both sides of the report.
- **Pivot:** changes one dimension from one side of the report to the other (rows by columns).

The most important trait of these operators is that they are “query modifiers” (i.e., they are used to “refine”

Table 2. A hybrid data cube/table output of a multidimensional query

		2005-1	2005-2	2005-3	2005-4	2006-1	2006-2
Spain	Parfums	17000	12000	3000	11000	18000	12000
	Pet meat	57000	52000	53000	56000	54000	54000
	Fresh vegetables	93000	125000	131000	103000	97000	133000
	Pet accessories	22000	32000	23000	19000	24000	35000
	Alcoholic drinks	5000	4200	3800	7300	5300	4100
	Garden & furniture	12000	18000	19000	15000	14000	18000
France	Parfums	23000	17000	5000	17000	17000	13000
	Pet meat	77000	63000	72000	82000	64000	53000
	Fresh vegetables	102000	132000	126000	110000	99000	143000
	...	...	...	...	...	...	...
	...	...	...	...	...	...	...
	...	...	...	...	...	...	...
...	...	...	...	...	...	...	

the queries). The design of data warehouse and OLAP tools are intended to avoid the recalculation of the aggregates each time one of the previous operators is applied. The idea is to precalculate the aggregations at several levels of the hierarchies and so making the traversal through the dimension paths more efficient.

## Design and Implementation

One of the reasons to create a separate data repository for data analysis was to improve efficiency. The multidimensional model and the OLAP refinement operators are two powerful means towards this goal, but they must be complemented by an adequate data warehouse design and implementation.

There are, of course, many design methodologies for data warehouses (see e.g., Gardner, 1998; Inmon, 2002; Kimball, 1996; Sapia, Blaschka, Höfling, & Dinter 1996; Trujillo, Palomar, & Gómez, 2001; Tryfona, Busborg, & Christiansen, 1999). Some of them differ significantly from the rest, using entity-relationship extensions, UML extensions, or specific modeling paradigms. There are also international standards, such as the OMG (object management group) common warehouse metadata (CWM) for this purpose.

The following are some hints on the most important steps for designing a data warehouse (or more precisely, each data mart), under the multidimensional model:

- Choose a “process” or “domain” in the organization for which analytical processing is to be done, considering the data that complex reports, more general analysis, or data mining will require.
- Choose the central fact and the granularity. Granularity should be fine-grained enough to allow every interesting report and query to be done, but not more.

- Identify the dimensions that characterize the domain, its levels and hierarchies, and the basic attributes for each level. Dimensions can vary from one domain to the other, although they usually correspond to questions such as *what*, *who*, *where*, *when*, *how*, and so forth.
- Determine and refine the measures for the facts (usually corresponding to questions such as *how many* or *how much*) and the attributes for the dimensions.

Once the data warehouse is designed, it is time to start its implementation (Weir, Peng, & Kerridge, 2003). The first important decision for implementation is to determine the kind of physical environment that will hold the information. There are two main approaches:

- Relational OLAP (ROLAP). Physically, the data warehouse will be implemented with a relational database.
- Multidimensional OLAP (MOLAP). Physically, the data warehouse is constructed over special structures based on multidimensional matrices.

It can be argued that the MOLAP approach should be more adequate for a data warehouse that uses a multidimensional model. However, there are pros and cons for both approaches. ROLAP has several advantages: There are many RDBMS to choose from, with their typical tools (e.g., SQL, constraints, triggers, procedures); the training and cost required for the implementation of a ROLAP data warehouse is usually lower. MOLAP has a more direct correspondence between the logical level and the physical level, and it is usually more efficient.

At least in the inception of data warehouses, the ROLAP approach has been more successful. There are many methodologies to convert a multidimensional de-

sign into a relational database that implements it. The *starflake* structure (Kimball, 1996) is a well-known approach and is based on a central *fact table*, with foreign keys to base (fine-grained) dimension tables, which, in turn, refer to other, more coarse-grained dimension tables. These tables are called *snowflake tables*. Some additional tables (*star tables*) are added (redundantly) to improve efficiency.

Associated to the physical level, new technologies have been developed to make the data warehouse work effectively: tools to compute and instantiate the data cube, query optimization to select and refine queries on the data cube, specialized indexing techniques for improving aggregation, and many others.

### Data Load and Refreshment

Finally, there is an important component to be considered for the implementation of a data warehouse. Once we have chosen the data we want to integrate in the data warehouse (either internal or external), we have designed the schema that will be used to organize all this data, decided the implementation over ROLAP or MOLAP and created the physical schema, we can start with the hard and true implementation work. But what is left? We have to extract and transform the data from the original sources (and from several schemas) and to load them into the data warehouse (into the new schema).

Although this process may resemble some data migration processes, which are common when one or more databases have to be moved into a new and different system, there are some specific issues which make this process even more complicated. The volume of data transferred is usually very large, the data has to be transformed deeply and comes from many different sources, some original databases cannot be disturbed by this process because they are necessary for the daily transactional work, and, finally, the data has to be refreshed periodically (i.e., the data warehouse must be kept up to date, weekly or monthly, with the new data added to the transactional sources).

The complexity of this process and the fact that it must be operating periodically motivates the construction of a *system*, which is called the extraction, transformation, load (ETL) system.

The ETL system must perform many tasks:

- **Extraction of the transactional data:** This must be done in such a way that the transactional database is not disturbed (e.g., at night).
- **Incorporation of external data:** Since the data might be unstructured, it is necessary to use wrap-

pers or scripts to convert data from text, spreadsheets, HTML, or other formats into the form required for the data warehouse.

- **Key creation and metadata creation:** The data moved must be assigned new keys. The correspondence between the original data and the data in the warehouse must be traceable through the use of metadata.
- **Data integration, cleansing, transformation, and aggregation:** The data must be crossed, compared, and integrated, and most of the data is needed in some level of aggregation. It is not convenient to perform all this on the transactional databases, because this can slow the transactional work.
- **Change identification:** The system must deal about how changes are refreshed from the sources to the data warehouse (e.g., by delta files, time stamping, triggers, log files, mixed techniques).
- **Load planning and maintenance:** Definition of the order and stages for the load process, load time windows (to minimize impact on the transactional sources), creation of indexes on the data warehouse, and so forth. The maintenance must include periodical assessments on the quality of the data stored in the data warehouse.

Many ETL systems use an intermediate repository between the sources and the data warehouse to be able to do aggregations, integration, and cleansing, as well as other processes (metadata) without disturbing the original transactional databases or the destination data warehouse.

### FUTURE TRENDS

Data warehousing technology is evolving quickly and there are many research opportunities (Dinter, Sapia, Hölfing, & Blaschka, 1999; Rizzi, 2003; Roddick, 1998; Rossopoulos, 1998; Samtani, Kumar, & Kambayashi, 1998; Widom, 1995) on many topics, such as data models, logical and physical design, implementation (the construction of the ETL system in particular), metadata standardization, theoretical issues (e.g., materialized views, schema versioning, evolution), as well as other, more heterogeneous topics, such as data cleansing and transformation, XML metadata, OLAP query languages, the operability and friendliness of OLAP tools, and their distribution between client and server.



## CONCLUSION

Data warehousing is an emerging technology that has spread over organizations and companies all over the world, because it is making OLAP possible (i.e., a real-time analytical processing for volumes of data, which was simply unbelievable a decade ago). The use of separate data repositories, called data warehouses, with the organization under specific data models and specific physical implementation, as well as new OLAP tools and operators have boosted the friendliness and effectiveness of reporting and other analytical tools. Data warehouses are not only used as platforms for query and report tools with an orientation to “farming” reports, but they also make a very good team with more “exploring” objectives, such as summarization and data mining.

## REFERENCES

- Dinter, B., Sapia, C., Höfling, G., & Blaschka, M. (1999) OLAP market and research: Initiating the cooperation. *Journal of Computer Science and Information Management*, 2(3), 23-37.
- Gardner, S. R. (1998) Building the data warehouse. *Communications of the ACM*, 41(9), 52-60.
- Golfarelli, M., Maio, D., & Rizzi, S. (1998) Dimensional fact model. *International Journal of Human-Computer Studies*, 43(5/6), 865-889.
- Inmon, W. H. (1992). EIS and the data warehouse: A simple approach to building an effective foundation for EIS. *Database Programming and Design*, 5(11), 70-73.
- Inmon, W. H. (2002). *Building the data Warehouse* (3rd Ed.). Chichester, UK: Wiley.
- Kimball, R. (1996) *The data warehouse toolkit: Practical techniques for building dimensional data warehouses*. New York: Wiley.
- Rizzi, S. (2003, September 8). Open problems in data warehousing: Eight years later (Keynote slides). In H.-J. Lenz, P. Vassiliadis, M. Jeusfeld, & M. Staudt (Eds.), *Design and management of data warehouses 2003. Proceedings of the 5th International Workshop*, Berlin, Germany. Available online from <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-77/keynote.pdf>
- Roddick, J. F. (1998). Data warehousing and data mining: Are we working on the right things? Lecture notes in computer science. In Y. Kambayashi, D. K. Lee, E.-P. Lim, Y. Masunaga, & M. Mohania (Eds.), *Advances in database technologies* (pp. 141-144). Berlin, Germany: Springer-Verlag.
- Rossopoulos, N. (1998). Materialized views and data warehouses. *SIGMOD Record*, 27(1), 21-26.
- Samtani, S., Kumar, V., & Kambayashi, Y. (1998) *Recent advances and research problems in data warehousing*. International Conference on Conceptual Modeling (ER), Singapore.
- Sapia, C., Blaschka, M., Höfling, G., & Dinter, B. (1998). *Extending the E/R model for the multidimensional paradigm*. International Workshop on Data Warehouse and Data Mining (DWDWM), Singapore.
- Trujillo, J. C., Palomar, M., & Gómez, J. (2001). Designing data warehouses with OO conceptual models. *IEEE Computer*, 34(12), 66-75.
- Tryfona, N., Busborg, F., & Christiansen, J. (1999). Star ER: A conceptual model for data warehouse design. *International Workshop on Data Warehousing and OLAP (DOLAP99)*, Kansas City, Missouri.
- Weir, R., Peng, T., & Kerridge, J. (2003, September 8). Best practice for implementing a data warehouse: A review for strategic alignment. In H.-J. Lenz, P. Vassiliadis, M. Jeusfeld, & M. Staudt (Eds.), *Design and Management of Data Warehouses, Proceedings of the 5th International Workshop*, Berlin, Germany.
- Widom, J. (1995). Research problems in data warehousing. *International Conference on Information and Knowledge Management*.

## KEY TERMS

**Data Cube:** A preselection of aggregated data at several levels of detail according to several dimensions in a data mart. A data cube establishes the resolution and basic projection where selections can be made. The data cube aggregation detail can be modified through OLAP operators such as drill and roll, and their dimensions can be modified by slice & dice and pivot.

**Data Mart:** Part of a data warehouse which gathers the information about a specific domain. Each data mart is usually viewed at the conceptual model as a multidimensional star or snowflake schema. Although each data mart is specialized on part of the organization information, some dimensions can be redundantly replicated in several data marts. For instance, time is usually a dimension shared by all data marts.



**ETL (Extraction, Transformation, Load) System:** The system which is in charge of extracting the data from internal transactional databases and other sources (e.g., external data), transforming it to accommodate to the data warehouse schema, loading the data initially and refreshing the data periodically. The design of the ETL system is generally the most time-consuming task in the construction of a data warehouse.

**Farmers and Explorers:** Two typologies of data warehouse users. Farmers are users of data warehouses and other analytical tools that generate periodical reports, such as sales by week, category, and department. Explorers are more ambitious users of data warehouses which try to better understand the data, to look for patterns in the data or to generate new reports.

**MOLAP (Multidimensional OLAP):** Implementation of a data warehouse and associated OLAP tools over specific multidimensional data structures. This approach has a more direct mapping between the multidimensional conceptual schema and the multidimensional physical schema.

**OLAP (On-Line Analytical Processing):** The process of analyzing a database or data warehouse, which

consists of *heavy* queries for constructing reports or showing the information in a highly aggregated or complex way. This kind of processing supports information retrieval and data analysis in the form of complex queries and reports.

**OLTP (On-Line Transactional Processing):** The usual process in a transactional database, which consists of frequent queries and updates for serving the applications of an organization. This kind of processing supports the daily operations of the software applications, it is read-and-write, and generally performed through SQL queries.

**ROLAP (Relational OLAP):** Implementation of a data warehouse and associated OLAP tools over a relational database and RDBMS. This approach must be based on an appropriate mapping between the multidimensional conceptual schema and the relational logical/physical schema. The great advantage of this approach is that inherits the existing technology and knowledge on classical relational database systems.

**Snowflake Schema:** A schema for organizing multidimensional data, where we have many levels in each dimension and there can be alternative paths for aggregation. This schema is the most usual for designing data marts.

# Data Warehousing, Multi-Dimensional Data Models and OLAP

**Prasad M. Deshpande**

*IBM Almaden Research Center, USA*

**Karthikeyan Ramasamy**

*Juniper Networks, USA*

## INTRODUCTION

Since the advent of information technology, businesses have been collecting vast amounts of data about their daily transactions. For example, a company keeps track of data regarding the sales of its various products at different stores over a period of time. Businesses can gain valuable insights by analyzing this data to spot trends and correlations in the data. Data warehousing, multidimensional analysis, and online analytical processing (OLAP) refer to a set of technologies that address the problem of business data analysis. Data warehousing has become the established paradigm for knowledge workers to sift through mountains of historical data in order to extract nuggets of business information. Data analysis tools have been used in various forms historically since the 1960s (Pendse, 2003). Recently, there has been a rapid growth in the industry, with the total worldwide OLAP market estimated at about \$3.7 billion in 2003 (Pendse). This has also been an active area of research with many contributions in data warehouse design, storage, view selection, cube computation, indexing, and query evaluation.

## BACKGROUND

A data warehouse is a copy of transaction data specifically structured for querying and analysis (Kimball, 1996). Businesses want to separate their operational data from the data used for analysis for various reasons such as performance, maintenance, and security. Operational data is optimized for online transaction processing (OLTP) transactions that include lots of updates and simple queries. Analysis queries, on the other hand, are typically read only and need to aggregate large amounts of data. Data warehousing refers to the process of transforming the operational data, removing the extra fields that are not necessary for analysis, and storing it in a specialized store. In a typical scenario, the data in a data warehouse need not be current and is updated peri-

odically with fresh data from the main store. This flexibility makes it possible to optimize the data warehouse for analytical queries rather than for updates. The data stored in a warehouse is typically multidimensional in nature and consists of a set of dimensions and metrics. In our sales example, the dimensions are products, stores, and time. The data records the sales value for different combinations of these dimensions. “Sales” is referred to as a metric in this case. Analysts might ask queries such as:

- Q1. What are the total sales for each product?
- Q2. What are the sales for the product “Skis” in the month of January over all stores?

To answer these queries, we would have to aggregate the data in different ways. For example, Q1 requires the addition of sales figures across all stores and over all time for each product. Multidimensional data analysis refers to this process of viewing and analyzing the data along different dimensions. A multidimensional data model is a multidimensional view of the data that identifies the various dimensions and metrics in it. Dimensions often have hierarchies on them along which they can be analyzed. For example, the time dimension has the hierarchy of time, date, month, quarter, and year. We could ask queries at different levels in the hierarchy, such as the monthly sales, quarterly sales, and yearly sales.

OLAP tools provide multidimensional analysis functionality such as aggregation, forecasting, ranking, rolling up, drilling down, etc. Though there is no standard definition of OLAP, it can be summarized in five keywords: *fast analysis of shared multidimensional information* (Pendse, 2003). The critical challenge for OLAP tools is to make the analysis fast so that it can be interactive. The result of the queries might be small, but the queries are quite expensive to compute since they aggregate a lot of data to get the results. For example, for Q1 we need to aggregate the sales data across all stores and over all time, which can be huge. OLAP systems use a wide variety of techniques such as specialized storage and indexing, pre-computation, and caching to provide interactive analysis.

## CURRENT TECHNOLOGIES

Research in data warehousing and OLAP can be classified into the following broad areas.

### Data Warehouse Design

A data warehouse is a collection of integrated information that has been designed for analytical querying. The source of this data could be one or more operational OLTP systems or other legacy systems. To get the data in a usable form, the source data needs to be processed to remove the unnecessary attributes, aggregated, and mapped to the schema of the data warehouse. The major issues in building a data warehouse are schema design and maintenance issues such as data cleansing, initial loading, incremental updates, data purging, and metadata management. The warehouse schema is designed by first building a conceptual schema in terms of dimensions and metrics and then mapping it to a logical schema that might be relational or multidimensional. The methodologies proposed for schema integration and multidatabase systems can be applied for data cleansing and mapping the source data to the destination schema. A data warehouse can be considered to be a set of materialized views on the source data. The updates in the source data need to be propagated to the data warehouse. There are various issues to be considered such as consistency requirements, refreshing timing, refreshing frequency, refreshing modes, and refreshing techniques. OLAP systems typically don't have as stringent requirements as OLTP systems regarding the currency of data. Thus most systems update the data warehouse on a periodic basis. The updates could be done in an online fashion with the warehouse still running or an off-line fashion by shutting down the system during the refresh period. Updating the data also requires updating of the various auxiliary structures such as indices and pre-computed aggregates that might be built on it. There has been a lot of research on the problem of view maintenance; we cite only a representative sample (Labio, Yerneni, & Garcia-Molina, 1999; Mumick, Quass, & Mumick, 1997; Quass & Widom, 1997). Metadata management plays a key role in data warehouses for keeping track of what data is available, where it is available, mapping between the different data elements, transformations required, etc. Jarke, Lenzerini, Vassiliou, and Vassiliadis (2003) stress that the quality of a data warehouse can be accessed and improved using enriched quality-focused metadata management and methodologies.

## Data Modeling And Storage

The data in a data warehouse is multidimensional in nature. Though this data could be modeled in traditional ways such as ER modeling or relational modeling, it is more intuitive to think of it in terms of dimensions and facts. Facts represent the entity being measured and are a function of the dimension values. In our sales example, Product, Store, and Time are dimensions, and Sales is a metric. Typically the values for a dimension can be grouped in a hierarchical tree structure. For example, the hierarchy for the time dimension is shown in Figure 1. The analyst can view data at different levels along the hierarchy. Different aggregation functions can be applied to the lower-level data in order to get data at higher levels. Some commonly used aggregation functions are sum, count, min, max, and average. For example, the analyst might want to see the total sales figure at a monthly level. In this case, the aggregate function "sum" will be applied to sales values within each month to get the monthly figure. Some dimensions can have multiple hierarchies defined on them. For example, another hierarchy on the time dimension is week, quarter, year.

The multidimensional data can be stored and organized in different ways. There are two contrasting approaches to this called relational OLAP (ROLAP) and multidimensional OLAP (MOLAP). In a ROLAP system, the data is stored in standard relational tables. The analytical engine is built on top of the relational system and uses standard SQL to access data in the tables. The multidimensional data model is most commonly mapped onto a star schema. A star schema consists of a set of fact tables and dimension tables. A fact table has measure attributes that record the facts and dimension attributes that form a foreign key to the dimension tables. The dimension tables store the dimension values, their attributes, and the hierarchies on them. The star schema for the sales example consists of a fact table called Sales and three dimension tables called Product, Time, and Store as shown in Figure 2.

Figure 1. Hierarchy in Time dimension

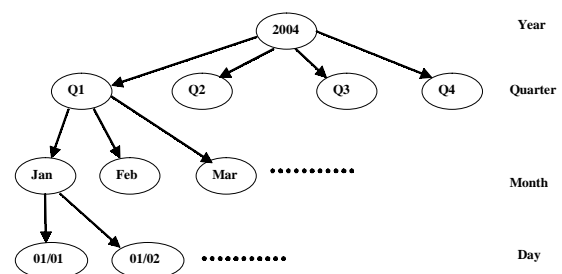
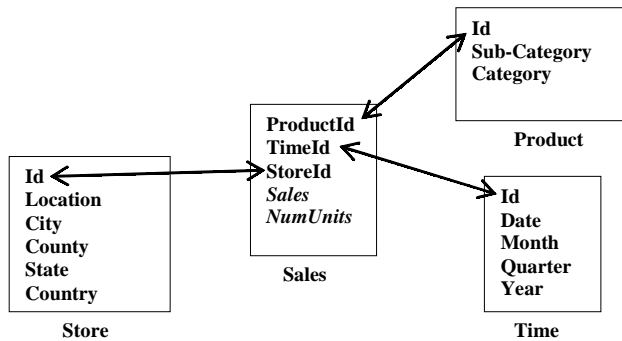


Figure 2. Star Schema



The dimension tables in a star schema are de-normalized in order to reduce the number of joins during query processing. The alternative approach is to have a snowflake scheme in which each dimension table is normalized into multiple tables. The advantages of ROLAP are that it uses tested relational technology with well-developed query processing and optimization, is highly scalable, can coexist with other data sources, and does not need any specialized storage mechanisms.

MOLAP systems, on the other hand, build a specialized data store that allows multidimensional access to the data. The data could be stored as multidimensional arrays, but specialized techniques are necessary to handle large arrays and sparsity of data. A lot of research has been done in developing specialized multidimensional storage and the related query processing (Cheung, Zhou, Kao, Kan, & Lee, 2001; Deshpande, Ramasamy, Shukla, & Naughton, 1998; Sarawagi & Stonebraker, 1994; Zhao, Deshpande, & Naughton, 1997). MOLAP systems typically have better performance due to specialized storage structures, but the downside is the lack of scalability, size blowup, and the need to re-implement storage and query processing.

## View Selection and Size Estimation

In order to efficiently answer queries, many OLAP systems pre-compute aggregates/views along some dimensions and store them in the data warehouse. However, the number of possible aggregates is exponential in the number of dimensions. Due to space and time limitations, it is impossible to compute all the possible aggregates. Under these constraints, it is important to choose a set of aggregates to pre-compute in order to reduce the overall query execution time. A lot of algorithms have been proposed for this problem, including a greedy optimization algorithm (Harinarayan, Rajaraman, & Ullman, 1996), a constrained lattice solution (Baralis, Paraboschi, & Teniente, 1997), a

simple size-based algorithm (Shukla, Deshpande, & Naughton, 1998), and a maintenance time-constrained solution (Gupta & Mumick, 1999). Since this choice is based on the sizes of the aggregates, a related problem is to estimate the sizes, for which statistical- and sampling-based methods have been proposed (Shukla, Deshpande, & Naughton, 1996). The data warehouse needs to be aware of the materialized views and exploit them during query processing. If any of the views can be used optimally, queries against the warehouse are rewritten to run against the pre-computed views rather than the base data (Halevy, 2001; Park, Kim, & Lee, 2001; Srivastava, Dar, Jagadish, & Levy, 1996).

## Query Processing

OLAP operations correspond to exploring the multidimensional data space by moving up the dimension hierarchy (roll up), moving down (drill down), restricting to a dimension value (slice), selecting an aggregated subspace (dice), and cross tabulation (pivot). The basic requirement to answer all these queries is aggregation. OLAP systems need specialized query processing and indexing techniques to handle complex aggregate queries efficiently. Depending on the architecture (ROLAP or MOLAP), different techniques can be used. For relational systems, the SQL language has been extended to be able to specify OLAP kinds of operations. One of the earliest extensions was the CUBE operator proposed by Gray et al. (1997) in order to specify a set of group-bys that are computed together. SQL extensions involving array-based calculations for complex modeling and spreadsheets have also been recently proposed (Witkowski, 2003). In the multidimensional world, there have been efforts to define a query language specialized for multidimensional data such as the MDX language (Whitehorn, Zare, & Pasumansy, 2003).

One of the important problems in OLAP is computing and organizing the cube. The cube is a set of all possible aggregates over the dimensions. For example, a Sales cube on the Store and Product dimensions will have the following group-bys: Sales by (Store, Product), Sales by (Store), Sales by (Product), and the total Sales. Most of the techniques of computing the cube try to share the common computations among the different group-bys in order to reduce the overall time (Agarwal et al., 1996; Ross & Srivastava, 1997; Zhao et al., 1997). Since a cube can be very big, various ways to restrict the cube have been proposed such as the iceberg cube that considers only aggregate values above a certain threshold (Beyer & Ramakrishnan, 1999; Fang, Shivakumar, Garcia-Molina, Motwani, & Ullman, 1998; Xin, Han, Li, & Wah, 2003). Some systems consider reducing the size by computing compressed cubes by approximation, building special-

ized indices, or other ways of condensing the cube (Barbara & Sullivan, 1997; Lakshmanan, Pei, & Han, 2003; Sismanis, Deligiannakis, Roussopoulos, & Kotidis, 2002). Another interesting class of queries in OLAP is the top-k queries that return k tuples having the highest or lowest values for some attribute, expression, or function (Carey & Kossmann, 1997). In ROLAP systems, the join between the fact table and the dimension tables is a frequent operation that can be quite expensive. Join indices and bit-map indices have been proposed to make this operation more efficient (Chan & Ioannidis, 1998). Many OLAP systems build a mid-tier cache to exploit the locality in OLAP queries and reuse results. Several architectures for building a multidimensional cache specialized for OLAP have been proposed in the literature (Dar, Franklin, Jonsson, Srivastava, & Tan, 1996; Deshpande et al., 1998; Scheuermann, Shim, & Vingralek, 1996;).

## **FUTURE TRENDS**

Typically, OLAP works on data that is periodically refreshed. However, in some cases, currency of data might be a critical requirement. Providing real-time decision support in such a case is still a challenge. Also, traditionally, OLAP has dealt with structured data. However, there is a lot of data that is unstructured, in the form of text fields and documents. Analyzing this text data to infer meaning and sentiment opens up new problems. There is work in probabilistic OLAP aimed at providing an OLAP kind of analysis on a combination of structured and unstructured data.

## **CONCLUSION**

Data warehousing and OLAP are becoming increasingly important for business intelligence in order to extract key insights from the vast amounts of data being collected. Fast analysis is the goal for these systems so that users can interactively analyze the data. There are a lot of challenges in building such a system, including data modeling, schema design, loading, maintenance, query processing, etc. After a late start compared to the industry, there has been a lot of active research in this area contributing key new technologies. Among the OLAP vendors there are two main camps: those that build a specialized multidimensional engine and those trying to push a lot of OLAP functionality into relational databases. Both approaches have their merits, and the end result might just be a hybrid system in which a multidimensional middle tier sits on a relational back-end.

## **REFERENCES**

- Agarwal, S., Agrawal, R., Deshpande, P. M., Gupta, A., Naughton, J. F., Ramakrishnan, R., et al. (1996). On the computation of multidimensional aggregates. *Proceedings of the 22<sup>nd</sup> International Conference on VLDB* (pp. 506-521).
- Baralis, E., Paraboschi, S., & Teniente, E. (1997). Materialized view selection in a multidimensional database. *Proceedings of the 23<sup>rd</sup> International Conference on VLDB* (pp. 156-165).
- Barbara, D., & Sullivan, M. (1997). Quasi-cube: Exploiting approximation in multidimensional databases. *SIGMOD Record*, 26, 12-17.
- Beyer, K., & Ramakrishnan, R. (1999). Bottom-up computation of sparse and iceberg cubes. *Proceedings of the 1999 ACM SIGMOD Conference* (pp. 359-370).
- Carey, M. J., & Kossmann, D. (1997). Processing top N and bottom N queries. *IEEE Data Engineering Bulletin*, 20, 12-19.
- Chan, C. Y., & Ioannidis, Y. E. (1998). Bitmap index design and evaluation. *Proceedings of the 1998 ACM SIGMOD Conference* (pp. 355-366).
- Cheung, D. W., Zhou, B., Kao, B., Kan, H., & Lee, S. D. (2001). Towards the building of a dense-region-based OLAP system. *Data and Knowledge Engineering*, 36(1), 1-27.
- Dar, S., Franklin, M. J., Jonsson, B. T., Srivastava, D., & Tan, M. (1996). Semantic data caching and replacement. *Proceedings of the 22<sup>nd</sup> International Conference on VLDB* (pp. 330-341).
- Deshpande, P. M., Ramasamy, K., Shukla, A., & Naughton, J. F. (1998). Caching multidimensional queries using chunks. *Proceedings of the 1998 ACM SIGMOD Conference* (pp. 259-270).
- Fang, M., Shivakumar, N., Garcia-Molina, H., Motwani, R., & Ullman, J. (1998). Computing iceberg queries efficiently. *Proceedings of the 24<sup>th</sup> International Conference on VLDB* (pp. 299-310).
- Gupta, H., & Mumick, I. S. (1999). Selection of views to materialize under a maintenance cost constraint. *ICDT '99, Seventh International Conference* (pp. 453-470).
- Gray, J., Chaudhari, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., & Pirahesh, H. (1997). Data Cube: A Relational Aggregation Operator Generalizing Group-by, Cross-Tab, and Sub Totals. *Data Mining and Knowledge Discovery*, 1, 29-53.



- Halevy, A. Y. (2001). Answering queries using views: A survey. *The VLDB Journal*, 10, 270-294.
- Harinarayan, V., Rajaraman, A., & Ullman, J. D. (1996). Implementing data cubes efficiently. *Proceedings of the 1996 ACM SIGMOD Conference* (pp. 205-216).
- Jarke, M., Lenzerini, M., Vassiliou, Y., & Vassiliadis, P. (2003). *Fundamentals of data warehouses* (2nd rev. ed.). New York: Springer-Verlag.
- Kimball, R. (1996). *The data warehouse toolkit: Practical techniques for building dimensional data warehouses*. Hoboken, NJ: Wiley, John & Sons.
- Labio, W. J., Yerneni, R., & Garcia-Molina, H. (1999). Shrinking the warehouse update window. *Proceedings of the 1999 ACM SIGMOD Conference* (pp. 383-394).
- Lakshmanan, L. V. S., Pei, J., & Han, J. (2002). Quotient cubes: How to summarize the semantics of a data cube. *Proceedings of the 28<sup>th</sup> International Conference on VLDB* (pp. 778-789).
- Mumick, I. S., Quass, D., & Mumick, B. S. (1997). Maintenance of data cubes and summary tables in a warehouse. *Proceedings of the 1997 ACM SIGMOD Conference* (pp. 100-111).
- Park, C.-S., Kim, M. H., & Lee, Y.-J. (2001). Rewriting OLAP queries using materialized views and dimension hierarchies in data warehouses. *Proceedings of the 17th International Conference on Data Engineering* (pp. 515-523).
- Pendse, N. (2003). *The OLAP report*. Retrieved from <http://www.olapreport.com/>
- Quass, D., & Widom, J. (1997). On-line warehouse view maintenance. *Proceedings of the 1997 ACM SIGMOD Conference* (pp. 393-404).
- Ross, K., & Srivastava, D. (1997). Fast computation of sparse datacubes. *Proceedings of the 23rd International Conference on VLDB* (pp. 116-125).
- Sarawagi, S., & Stonebraker, M. (1994). Efficient organization of large multidimensional arrays. *Proceedings of the 10th International Conference on Data Engineering* (pp. 328-336).
- Scheuermann, P., Shim, J., & Vingralek, R. (1996). Watchman: A data warehouse intelligent cache manager. *Proceedings of the 22nd International Conference on VLDB* (pp. 51-52).
- Shukla, A., Deshpande, P. M., Naughton, J. F., & Ramasamy, K. (1996). Storage estimation for multidimensional aggregates in the presence of hierarchies. *Proceedings of the 22nd International Conference on VLDB* (pp. 522-531).
- Shukla, A., Deshpande, P. M., & Naughton, J. F. (1998). Materialized view selection for multidimensional datasets. *Proceedings of the 24th International Conference on VLDB* (pp. 488-499).
- Sismanis, Y., Deligiannakis, A., Roussopoulos, N., & Kotidis, Y. (2002). Dwarf: Shrinking the petacube. *Proceedings of the 2002 ACM SIGMOD Conference* (pp. 464-475).
- Srivastava, D., Dar, S., Jagadish, H. V., & Levy, A. Y. (1996). Answering queries with aggregation using views. *Proceedings of the 22nd International Conference on VLDB* (pp. 318-329).
- Whitehorn, M., Zare, R., & Pasumansy, M. (2003). *Fast track to MDX*. New York: Springer-Verlag.
- Witkowski, A., Bellamkonda, S., Bozkaya, T., Dorman, G., Folkert, N., Gupta, A., Sheng, L., & Subramanian, S. (2003). Spreadsheets in RDBMS for OLAP. *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data* (pp. 9-12).
- Wu, M. C., & Buchmann, A. P. (1997). Research issues in data warehousing. *Datenbanksysteme in Büro, Technik und Wissenschaft (BTW), GI-Fachtagung, Ulm, 5.-7. März 1997, Proceedings* (pp. 61-82).
- Xin, D., Han, J., Li, X., & Wah, B. W. (2003). Star-cubing: Computing iceberg cubes by top-down and bottom-up integration. *Proceedings of the 29th International Conference on VLDB* (pp. 476-487).
- Zhao, Y., Deshpande, P. M., & Naughton, J. F. (1997). An array-based algorithm for simultaneous multidimensional aggregates. *Proceedings of the 1997 ACM SIGMOD Conference* (pp. 159-170).

## KEY TERMS

**Aggregate Function:** A function that is used to aggregate a set of values to get a single value.

**Cube:** A collection of data aggregated at all possible levels over the dimensions.

**Dice:** Selecting a range on multiple dimensions to select a sub-cube of the original space.

**Dimension:** This refers to an axis along which the facts are recorded.

**Dimension Table:** A database table that stores the different possible values for a dimension, the attributes for those values, and the hierarchical values to which it maps.

**Dense Data:** Data that has metric values for a substantial percentage of all possible combinations of the dimension values.

**Drill Down:** The process of moving down a level along a dimension hierarchy during analysis. Drill down results in detailed data at a finer level of granularity.

**Fact Table:** A database table that stores the base facts that consist of values of different metrics for different combinations of dimension values.

**Hierarchy:** A hierarchy on a dimension represents the different levels at which data can be aggregated and viewed at along that dimension.

**Hybrid OLAP:** A hybrid architecture that stores the base data in a relational form, but also builds specialized persistent multidimensional structures on it to efficiently answer multidimensional queries.

**Metric/Measure:** A value that represents a certain fact that is being recorded in the transaction.

**MOLAP:** Architecture for OLAP systems that uses specialized multidimensional storage and access methods, such as arrays, for the data.

**Pre-Computation:** This refers to pre-computing commonly required aggregates that are expensive to compute on the fly.

**ROLAP:** Architecture for OLAP systems that uses a relational system as the storage mechanism for the data.

**Roll Up:** The process of moving up a level along a dimension hierarchy during analysis. Roll up results in more aggregated data at a coarser level of granularity.

**Slice:** Restricting the area of interest to a single value along a given dimension.

**Snowflake Schema:** A database schema with a fact table and a set of dimension tables. There are multiple tables per dimension with each table storing one level in the dimension hierarchy.

**Sparse Data:** Data that has metric values for a very small percentage of all possible combinations of the dimension values.

**Star Schema:** A database schema that consists of a single fact table and a single table per dimension. The dimension table stores the entire hierarchy for that dimension.

# Database Engineering Focusing on Modern Dynamism Crises

**Luiz Camolesi, Jr.**

*Methodist University of Piracicaba, Brazil*

**Marina Teresa Pires Vieira**

*Methodist University of Piracicaba, Brazil*

## INTRODUCTION

Researchers in several areas (sociology, philosophy, and psychology), among them Herbert Spencer and Abraham Maslow, attribute human actions resulting in continual environmental changes to the search for the satisfaction of individual and collective needs. In other fields of science, this behavior represents a challenge in ethical researches on concepts, methodologies, and technologies aimed at optimizing and qualifying the actions involved in these continual changes to obtain better results.

Specifically in computer science, software engineering is a critical sub-area for these researches and their application (Lehman & Stenning, 1997), since it involves the construction of models and orientation for their use in the development of resources, such as software, to support the user's needs (Perry & Staudenmayer, 1994). Databases are included in this context as a component for data storage.

Considering the premise of continuous changes (Table 1) and the human needs involved (Khan & Khang, 2004), the consequences for software and for the database used are obvious. In the field of computational science, these changes in the modern world are reflected in evolutionary features for software and databases (Brereton, Budgen & Bennet, 1999), based on database concepts, structures, and processes that allow for rapid, albeit not traumatic, shifts to new industrial, commercial, or scientific systems (Mcfadden, Hoffer & Prescott, 1999) in new contexts (*temporal scenarios*) (Camolesi, 2004).

Table 1. Types of changes

**Evolution:** actions for an (*variant*) element's technological progress, improvement, modernization, or correction.

**Revolution:** alteration actions of an element can influence the element's purpose in the context.

**Involution:** simplification actions of an element, regression in its conception or content.

## BACKGROUND

Database models must comprise representation elements that are adaptable to the user's varying and dynamic needs, and contain the taxonomy needed for their manipulation. Thus, traditional (generic) database models such as the Entity-Relationship (ERM) and Relational (RM) models (Siau, 2004) have been expanded with appropriate "profile" for specific applications and requirements. Considering their purpose of supporting changes, "profile models" can be easily referenced in scientific researches as:

- **Version Model:** considering versions as database objects derived (originating from, but containing alterations) from others, models of this profile must be applied to a database characterized by the explicit and voluntary storage of the historical information about object changes (Conradi & Westfechtel, 1998). The features frequently specified in versions models are:
  - **Derivation Structure:** establishes the data structure for organizing versions, for example, stack, tree, or not cyclic digraph, linked by special relationships representing linear or nonlinear derivation actions;
  - **Versionable Element:** establishes which *variant* elements (database objects) can have versions created and represented in the database;
  - **Property of Versioning:** this is a feature that serves to define a versionable element. This

property must be dynamically established for each element during either its creation or its definition;

- **Version Status:** status set (state or situation) for the versions;
  - **Manipulation of Versions:** the creation, update, and deletion of versions can be accomplished implicitly either by the database system or by the user through specific command language;
  - **Operation Restrictions:** the manipulation of versions by users can be granted unconditionally, or restrictions may be imposed for each operation (create, update, and delete).
- **Time Model:** in models of this profile, the elements that represent the dimension time are established essentially to control the evolution of the database. The reliability of such time-based models depends on the unambiguous definition of temporal limits to be imposed in any business or scientific database system. In an evaluation of systems using the time representation, database designers find many variations and ambiguous representations that can degenerate the processing of the time value simply because they are ignorant of how time is represented, how it can be analyzed, or how it should be converted. Using a Time Model for the homogeneous representation of the data type *time* and *interval* enables the designer to improve the evolution control performance. Based on many researches about time representation and utilization (Allen, 1991; Bettini et al., 1998), the following features are identified for a homogeneous data type definition:
    - **Moment:** a time instant value;
    - **Granularity:** precision domain of time instant, this feature can be based on the ISO 8601 (International Organization for Standardization, 2000) standard, for example, PnYnMnDTnHnMnS, or any other standard established by the application;
    - **Orientation:** reference system for temporal representation, for example, Gregorian calendar (UTC or Coordinated Universal Time), Chinese calendar, Jewish calendar or others;
    - **Direction:** all orientation has a moment of origin (0), and a time may be the moment preceding or following this origin moment;
    - **Application:** specification of the use of the temporal representation, allowing for the semantic recognition of the type, independently of the context in which it is inserted. The at-

tribute *application* should indicate one value as: *Occurrence*: to specify a moment (time or interval) to carry through an action (either a past situation or a future one); *Duration*: using time or interval to specify the duration of an action (either a past situation or a future one); *Frequency*: to specify a moment (time or interval) used to record repetitions.

- **Configuration Model:** this “profile model” is based on and related to the version model, with version aggregations defined as *configurations* or *releases* (Conradi & Westfechtel, 1998). The *configurations* or *releases* are logical aggregation components or artifacts, selected and arranged to satisfy the needs of applications based on composition abstractions (Sabin & Weigel, 1998). Applications that require the composition of database objects are related with engineering, that is, Computer Aided Software Engineering (CASE) and Computer Aided Design (CAD);
- **Integrity Model:** required in all data models (RM, OOM, and ORM), elements are established in models of this profile to support data consistency and integrity. In certain typical databases, the number of constraints and actions for database integrity may involve hundreds of elements, which require periodical reviews since they are strictly related to continually changing real situations. The elements in these models vary in form and purpose, the most common being rules, *business rules* (Date, 2000) and *database constraints* (Doorn & Rivero, 2002; OMG, 2003);
- **User Model:** necessary in all data models (RM, OOM, and ORM), the elements that represent the users are established in models of this profile. The modeling of user features is critical in the evolution of a database because it involves a diversity of needs and changes in the operational behavior (insert, delete, alter, and select) in the database. The modeling of human behavior to design access privileges is a well-consolidated analytical process (Middleton, Shadbolt & Roure, 2004). However, this process must take into account the static and dynamic aspects of people and their activities (Perry & Staudenmayer, 1994). Analyses based on a static approach define *User Roles* that are appropriate for traditional applications, whose activities change with relative infrequency. In dynamic applications, however, static definitions are insufficient, since the requirements call for temporary and specific activities that are necessary to support the dynamic definition of *User Roles*.

Database engineering involves database models and profile models applied in development methodologies (Elmasri & Navathe, 2003), that is, a collection of correlated techniques arranged in a logical order, with orientation rules for the materialization of an objective. The objective of a database engineering methodology should be the creation of an optimized database (based on ORM or OOM) flexible to changes implemented through well-executed engineering phases (elicitation of requirements, viability analysis, design, testing, and implementation), particularly in the design stages (conceptual, logical, and physical) in which database models and profile models are used.

The logical order of a methodology established by a database designing or engineering group should reflect the group's level of maturity and its knowledge of the work context, that is, the group's dedication to the design, which can be based on two distinct methodologies:

- **Sequential Engineering:** traditional methodology for database engineering in which the phases are executed linearly and can be based on the bottom-up approach, that is, from details of the data requirements (attributes) to the recognition of elements (entity or objects). In top-down sequential engineering, the elements are first identified and then refined in detail;
- **Concurrent Engineering:** methodologies initially created for conventional areas of engineering (mechanical and electrical), they establish the simultaneous development of a "product" through the cooperation of designing groups working separately but in constant communication and exchange of information (Carter & Baker, 1992). Concurrent (or Simultaneous) Engineering has been adapted to the broad and complex process of software engineering, but in this case, systems may be required to support the cooperative work among project groups, such as Groupware software, researched in the CSCW (Computer Support Cooperative Work) area.

The methodology may also depend on the stage of the database's life cycle (creation, implementation, maintenance). Database Maintenance supervised by Administration (DBA) is motivated solely by the system's degenerating performance resulting from the constant modernization (restructuring) of the database (Ponniah, 2003). If degeneration is not prevented, it can lead to increasing information access costs, rendering the use of a database unfeasible.

In some cases, the database maintenance operations are insufficient or not sufficiently adapted to maintain the database qualities. This is usually the case when maintenance

and documentation updates in a database have been neglected over a long period. Thus, evolution and change-oriented database engineering should be:

- **Reverse Engineering:** methodologies to recognize and represent the structural, functional, and behavioral aspects of an already operating database (Blaha & Premerlani, 1998). This mode of engineering focuses on the process steps needed to understand and represent a current database, and must include an analysis of the context in which the database was engineered. Depending on the case, the database should be treated as a "black box" to recognize its data, or physically analyzed to identify special data structures and storage solutions;
- **Reengineering:** methodologies for converting databases in obsolete environments to more modern technologies. Reengineering introduces techniques for the restudy and conversion of obsolete data to new realities and the resulting redesign of databases (Sage, 1995). Reengineering can comprise three phases: Reverse Engineering, Delta ( $\Delta$ ), and Forward Engineering. Reverse Engineering involves the abstract and clear definition of data representations. In the Delta phase, the designer executes modifications in database systems to incorporate new functionalities (positive  $\Delta$ ) or to reduce them (negative  $\Delta$ ) in order to accomplish complete or partial implementations. The third and last phase, Forward Engineering, refers to the normal development of database systems following the natural stages.

## MAIN THRUST

The profile models were created or inserted in Object-Oriented Models (OOM); however, the creation and adaptation of the Object-Oriented paradigm to traditional data models enabled profile models to be widely applied, for example, in Unified Model Language (UML) (Naiburg & Maksimchuk, 2002) and in Object-Relational Models (ORM). Although not described in the current literature as profile models, these models have well-defined purposes with a variable degree of flexibility in relation to the generic database models.

Though still incipient, the composition of these profile models for use in database engineering allows a database to be defined with features which are essential for its changeability, characteristic associated to quality features listed in Table 2, allowing rapid changes with the smallest possible impact on the database and its users.



Table 2. Features to evolution

<p><b>Traceability:</b> The monitoring is an essential task for efficiency tracing of the designer and team jobs, aiming at better evaluation of production (modeling and engineering). To this, the <i>influence</i> (OMG, 2001) among elements should be inserted in the project to allow database engineers to recognize automatically which are the consequences of the alterations accomplished in relation to all the elements of a system (Ramesh &amp; Jarke, 2001). These consequences can be characterized as inconsistencies of data, what demand the revision of all the reached elements, directly or indirectly, for the accomplished alterations. The correct definition of the influences allows optimizing the actions of inconsistencies verification and consequently the reduction of the maintenance costs.</p> <p><b>Flexibility:</b> Facility degree to changes with the smallest possible impact on the project and the database user. Models used and project accomplished must be capable to support new requirements (Domingues, Lloret &amp; Zapata, 2003).</p> <p><b>Portability:</b> The necessities to transfer the database to new environments (Operating Systems or DBMS), new technologies (programming language), or interfaces can be a demand for the evolution of a database. To this, the database modeling accomplished must be capable to support these changes with low impact.</p> <p><b>Compatibility:</b> Facility degree to insert new elements in database model. To this, the database modeling must obey the rigid standards of concepts and techniques to accept specifications or components of the same standard.</p>
--

Proving the importance of changeability, are the emergent frameworks and patterns searching to unify and to integrate the profile models in database engineering.

How should it be executed? What are the benefits? What is the cost of this process? Who will be affected?

These issues have motivated researches on emergent topics such as:

**FUTURE TRENDS**

Database development methodologies, models, and processes are constantly updated to support the changing requirements that occur during the recovery processes of requirements or maintenance (Paton & Diaz, 1999). With these scenarios, the status of traditional database design shifted from a complex software engineering phase to that of engineering (Roddick et al., 2000). Thus, database engineering today serves as the foundation for the development of data modeling adapted to frequent changes in dynamic requirements established by the user.

The recognizing of needs for Database Evolution is a more complex task (Table 3) that involves questions such as: What should be altered? When should it be executed?

- Database Administration Policy or Database Maintenance Policy (Moore et al., 2001) with the definition of data-driven rules to management of expired solutions and innovation of models;
- Aspect-Oriented Database Engineering with new perspectives on evolution based on separation of concerns related to evolution. The aspects modeling (Table 2) supports the non-functional properties of database and the identification of influences, using specific languages to represent profile models;
- Database Modeling Languages and Database Engineering Tools with resources to assure the Features to Evolution (Table 2) in database engineering, not forgetting that the choice of degree evolu-

Table 3. Database maintenance operations

<p><b>Correction</b> of the Database Schema in response to problems found during the database operation phase.</p> <p><b>Integration</b> of Database Schemas for the creation of a single database.</p> <p><b>Adaptation</b> of the Database Schema to the new elements resulting from new requirements.</p> <p><b>Updating</b> of the schema elements to conform the changing user requirements and business evolution.</p> <p><b>Refinement</b> of the elements of the Database Schema that were insufficiently detailed during the design phase.</p> <p><b>Incorporation</b> of several databases into a “Federated Database”, maintaining the independence of each individual database.</p>
---

tion depends on the maturity designer team and stability (or instability) of users' requirements.

## CONCLUSION

As can be seen from the above descriptions, Database Evolution is a vast subject under constant discussion and innovation, which explains why so many subjects require analysis, particularly those involving profile models. Despite specific researches on this theme, pragmatic interest in the process of Database Evolution is not usually reflected in these researches due to the lack of details or progress achieved.

The difficulties encountered in many researches may be attributed to the theme's complexity. The evolutionary process can be defined by many variations in form: voluntary or involuntary; explicit or implicit; temporal or atemporal; recorded or not in databases in historical form; executed through a simple update of information or involving the physical or logical restructuring of the database; following predefined semantic rules; originating from the user (through an interface) or from the DBMS (Database Management System).

Database Administrators (DBA) should clearly indicate that the process of changes in a database, regardless of the characteristics and techniques utilized, should maintain or expand the database's adaptability, flexibility, and efficiency (Domingues et al., 2003) to conform to new technologies and to the company's growth-related requirements, without neglecting the crucial problem of application adaptations (Hick & Hainaut, 2003).

## REFERENCES

- Allen, J.F. (1991). Time and time again. *International Journal of Intelligent Systems*, 6(4), 1-13.
- Bettini, C., Dyreson, C.E., Evans, W.S., Snodgrass, R.T., and Wang, X.S. (1998). A glossary of time granularity concepts. *Lecture Notes in Computer Science*, 1399, 406-413.
- Blaha, M., & Premerlani, W. (1998). *Object-oriented modeling and design for database applications*. Indianapolis, IN: Prentice Hall.
- Brereton, P., Budgen, D., Bennett, K., et al. (1999). The future of software. *Communication of ACM*, 42(12), 78-84.
- Camolesi, L. (2004). Survivability and applicability in database constraints: Temporal boundary to data integrity scenarios. *Proceedings of VIEEE International Con-*

*ference on Information Technology: Coding and Computing*, (Vol. 1, pp. 518-522).

Carter, D.E., & Baker, B.S. (1992). *CE: Concurrent engineering*. Boston: Addison-Wesley.

Conradi, R., & Westfechtel, B. (1998). Version models for software configuration management. *ACM Computing Surveys*, 30(2), 232-282.

Date, C.J. (2000). *What not how: The business rules approach to applications development*. Boston: Addison-Wesley.

Domingues, E., Lloret, J., & Zapata, M.A. (2003). Architecture for managing database evolution. *Lecture Notes in Computer Science*, 2784, 63-74.

Doorn, J.H., & Rivero, L.C. (2002). *Database integrity: Challenges and solutions*. Hershey, PA: Idea Group Publishing.

Elmasri, R., & Navathe S.B. (2003). *Fundamentals of database systems* (4th ed.). Boston: Addison-Wesley.

Hick, J.M., & Hainaut, J.L. (2003). Strategy for database application evolution: The DB-MAIN approach. *Lecture Notes in Computer Science*, 2813, 291-306.

International Organization for Standardization. (2000). *ISO 8601: Data elements and interchange formats - information interchange - representation of dates and times*. Technical Committee ISO/TC 154.

Khan, K., & Khang, Y. (2004). *Managing corporate information systems evolution and maintenance*. Hershey, PA: Idea Group Publishing.

Lehman, M.M., & Stenning, V. (1997). Laws of software evolution revisited. *Lecture Notes in Computer Science*, 1149, 108-124.

McFadden, F.R., Hoffer, J.A., & Prescott, M.B. (1999). *Modern database management*. Boston: Addison-Wesley.

Middleton, S.E., Shadbolt, N.R., & de Roure D.C. (2004). Ontological user profiling in recommender systems. *ACM Transactions on Information Systems*, 22(1), 54-88.

Moore, B., Ellesson, E., Strassner, J., & Westerinen, A. (2001). *Policy core information model*. Version 1 Specification, Internet Engineering Task Force.

Naiburg, E.J., & Maksimchuk, R.A. (2002). *UML for database design*. Boston: Addison-Wesley.

Object Management Group (OMG). (2001). *SPEM: Software process engineering metamodel*.

Object Management Group (OMG). (2003). *OCL: Object constraint language*. Version 2.0.

Paton, N.W., & Diaz, O. (1999). Active database systems. *ACM Computing Surveys*, 31(1), 63-103.

Perry, D.E., & Staudenmayer, N.A. (1994). People, organizations and process improvement. *IEEE Software*, 11(4), 36-45.

Ponniah, P. (2003). *Database design and development: An essential guide for IT professional*. Indianapolis, IN: Wiley-IEEE Press.

Ramesh, B., & Jarke, M. (2001). Toward reference models for requirements traceability. *IEEE Transaction on Software Engineering*, 27(1), 58-93.

Roddick, J.F., et. al. (2000). Evolution and change in data management: Issues and directions. *ACM SIGMOD Record*, 29(1), 21-25.

Sabin, D., & Weigel, R. (1998). Product configuration framework: A survey. *IEEE Intelligente Systems*, 13(4), 42-49.

Sage, A.P. (1995). Systems engineering and systems management for reengineering. *The Journal of Systems and Software*, 30(1/2), 3-25.

Siau, K. (2004). *Advanced topics in database research* (Vol. 3). Hershey, PA: Idea Group Publishing.

## KEY TERMS

**Business Rule:** Expression or statement that represents a restriction of data or operations in a business domain. A collection of Business Rules is a behavioral guide to support the Business Policy (organizational strategy). Business Rules can be implemented as Database Constraints, depending on the form and complexity of the restriction (Date, 2000).

**Configuration:** Collection of versions of different elements that make up a complex element. Versions in a configuration must be stable; that is, they cannot be altered but can be changed by a new version of the same element. Configurations are abstractions representing semantic relationships among elements. Configurations serve to establish the scope of an application (temporal, spatial, or user) (Sabin & Weigel, 1998). Revised Configurations consolidated by users can be defined as Releases. *Versions* of a release cannot be altered or changed by another version. Configurations defined as releases cannot be deleted because they are or were used. User identifiers, valid intervals for use, and constraints are some important items of information associated with releases in databases.

**Database Constraints:** Boolean functions associated with elements of a database used to evaluate the integrity of actions or data that are inserted, removed, or updated. A Database Constraint can be defined as a set of predicates  $P_1 \wedge P_2 \wedge \dots \wedge P_k$ , where each predicate possesses the form  $C_1 \theta C_2$ , and where  $C_1$  is an attribute,  $\theta$  is a comparison operator, and  $C_2$  is either an attribute or a constant (Camolesi, 2004). The specification of a database constraint can be formalized using the Object Constraint Language (OCL) (OMG, 2003).

**Database Maintenance Policy:** Policy can be defined as a plan or guide to constrain the actions executed by an individual or a group (Moore et al., 2001). The Database Maintenance Policy establishes rules for the action of database designers that are executed in response to intrinsic and constant maintenance-related alteration needs, based on the designers' empirical knowledge and work capacity. The definition of goals, implementation phases, results, and desired outcomes is information associated with every policy or subset of rules.

**Influence:** Type of dependency association to conceptually and logically represent interrelations among elements and to define their levels of involvement prior to a process of alteration of the data (OMG, 2001). Influence modeling is based on impact specifications of modifications in database objects during the engineering process.

**Temporal Scenario:** A concept that adequately represents and includes the characteristics typical of evolution and is used in adaptive, dynamic, or flexible systems. The role of temporal scenarios is to represent situations in which the database requirements have been or will be modified (Camolesi, 2004). Every scenario reaches a moment in time when its existence begins (*opening*). Starting from this moment, the actors begin to perform (*acting*). Scenarios may reach a moment in time when they cease to exist (*closing*). Scenarios can *open/close* several times (*recurrent scenarios*). The definition of the opening and closing moments is obligatory for the specification of the temporal existence of *temporary scenarios*. The exceptions to this rule are the *permanent scenarios*, which do not *close*, and the ones that do not *open*.

**Time and Interval (datatype):** Fundamental datatypes to establish the temporal reference in a database. Time is symbolized by real numbers, based on a sequence of representative values to meet the application (Allen, 1991). Interval is an aggregation of two time moments intended to delimit and characterize the interval (Allen, 1991). The interval may represent *continuous* or *discrete* time.

**User Role:** Collection of structural and behavioral characteristics (i.e., privileges) established by the DBA, DBMS, or applications for use in dynamic or static privileges. Dynamic user roles are typically employed in flexible and evolutionary systems. User profiling refers to typical knowledge-based modeling to recognize user roles, based on questionnaires and interviews to identify behavioral patterns (Middleton et al., 2004).

**Variant:** Applied to many structural elements of a database (class, table, constraint, etc.), it is used to define mutable elements or elements whose modification is highly probable. An invariant is the opposite of a variant, that is,

an element not involved in the evolution of the database. Variants of database objects or object properties can be modified and can generate versions.

**Version:** An alternative of a database element, with variations in structure, function, or behavior, used in the context of an application as a boundary of the work executed. A version can represent a state of an evolving element with variant and invariant properties (Conradi & Westfechtel, 1998). A collection of actions executed on a database element can generate an element version if it results in significant alterations of the element's characteristics.

# Database Query Personalization

Georgia Koutrika

University of Athens, Greece

## INTRODUCTION

Traditional database and information retrieval systems have followed a *query-based* information access paradigm (i.e., information is returned to the user on the basis of a query issued). As a result, users issuing the same query are provided with the same answer. With the advent of the World Wide Web and hand-held electronic devices such as palmtops and cellular phones, information access entered a new era. Increasing amounts of information become available to a growing mass of untrained lay users through various access media. A user searching Web-resident information may have to reformulate queries issued several times and sift through many results until a satisfactory, if any, answer is obtained. As purely query-driven approaches may be inappropriate in this context, the need for a shift towards a more user-centered information access paradigm arises. To this end, different approaches aim to the personalization of the overall user experience at different levels: content selection, content presentation, and user interaction. There is no generally accepted definition of personalization, so I adopt a broad one as follows:

- *Personalization* is the approach of providing an overall customized, individualized user experience by taking into account the needs, preferences, and particular characteristics of a user or group of users.

Focusing on the level of personalized content selection, several distinct lines of research exist. There are two broad categories: filter-based and personalized approaches.

- *Filter-based* approaches filter system responses on the basis of a user profile, storing long-term user interests. In particular, *information filtering* methods employ profiles comprised of keywords (Foltz & Dumais, 1992). An antagonism is a representative example of customized Web-based newspapers employing information filtering methods (Sakagami, Kamba, & Sugiura, 1997). Web-accessible databases employ *continuous queries* to allow users to obtain new results from the underlying collection or stream without issuing the

same query repeatedly (Chen, DeWitt, Tian, & Wang, 2000; Liu, Pu, & Tang, 1999). User preferences are also used for providing *recommendations* (Karypis, 2001; Shahabi, Banaei-Kashani, Chen, & McLeod, 2001), and for ranking search results (Glover, Lawrence, Birmingham & Lee Giles, 1999; Smyth, Bradley, & Rafter, 2002).

It is worth noting that query-based and filter-based approaches have been characterized as two sides of the same coin (Belkin & Croft, 1992). Filter-based approaches deal only partially with the problem of information overload. This problem still haunts Web searches in which users may issue various queries expressing different information needs.

- *Personalized* approaches are based on the observation that different people find different things relevant; therefore, they may expect different answers to the same query. Consider a simple example: John and Ann access a Web-based movies database, searching for comedies. John is a fan of director W. Allen, and Ann is not. Traditional query-based systems would consider only the query issued and return the same, exhaustive list of comedies to both users. Focusing on the user enables a shift from what is called *consensus relevancy* in which the computed relevancy for the entire population is presumed relevant for each user, toward *personal relevancy* in which relevancy is computed based on each individual's characteristics (Pitkow et al., 2002). Personal agents and query personalization approaches belong here. *Personal agents* represent virtual assistants that learn user preferences by dialoguing in natural language with customers. These preferences are used for the formulation of personalized queries and generation of individual recommendations (André & Rist, 2002; Semeraro, Degemmis, Lops, Thiel, & L'Abbate, 2003). *Query personalization* approaches exploit user preferences stored in profiles to dynamically enhance *any* user query by integrating preferences relevant to it (Koutrika & Ioannidis, 2004b; Pitkow et al., 2002). Personalized results may be ranked according to user preferences.



Figure 1. Information access paradigms

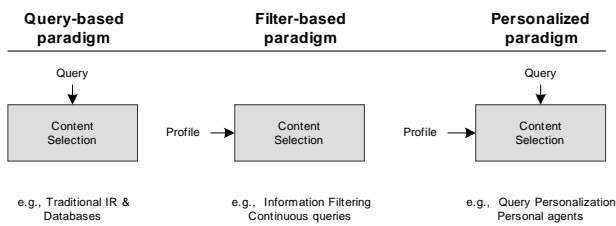


Figure 1 summarizes the aforementioned information access paradigms.

## BACKGROUND

Storing user preferences in user profiles gives a system the opportunity to return more focused personalized (and hopefully smaller) answers. The primary ways to personalize a search for an active searcher are *query augmentation* and *result ranking*. Returning to the previous example, John’s personalized results would include W. Allen’s comedies, and Ann’s would not. Which preferences are relevant to a specific request and how they affect the final answer are dynamically determined based on the query, the profile, and the personalization philosophy adopted. For example, when Ann is searching for a theatre to go to, the system should also consider her preference for downtown theatres. Query personalization approaches have recently attracted interest in both information retrieval and databases research communities.

*Outride* is a personalized IR system (Pitkow et al., 2002) that exploits user profiles defined upon the ontology of the open directory project (ODP). For query augmentation, the similarity between the query term and the user model is computed to decide which, if any, of the stored keywords are relevant to and should be included in the query. For example, if an individual is interested in coffee information and then searches for java, the system may dynamically augment the query by adding the relevant term *coffee* to provide the user with results about Java coffee. If another user interested in programming searches for java, the system may augment the query by adding the term *programming* or *language* to provide this user with results about the Java programming language. To rank search results based upon the user profile, metadata from the pages are compared via vector methods against the profile.

Another personalized IR system is described by Liu, Yu, and Meng (2002). A query is mapped to a set of categories stored in a user profile. A category is comprised of a set of terms with weights. A term weight

reflects the significance of the term in representing the user’s interest in that category.

Koutrika and Ioannidis (2004b) have implemented a first personalized database system and provide a query personalization framework that specifies which preferences from a profile should affect a query issued. Based on this framework, the top-*K* preferences are integrated into the query producing one that returns results satisfying at least *L* of them. Preferences are stored as *degrees of interest* in atomic query elements. Consider a user that accesses a movies database looking for films released in 2004. The corresponding SQL query could look like this:

```
Select title from MOVIES where year = 2004
```

Assuming the user is interested in comedies, the system may execute a personalized query that could look like this:

```
Select title from MOVIES, GENRES
where MOVIES.id=GENRES.id
and year = 2004 and genre = “comedy”
```

The aforementioned efforts have shown that the benefits of personalized search can be significant, appreciably decreasing the time it takes people to find information. In the following section, I focus on personalization of database queries.

## DATABASE QUERY PERSONALIZATION

A personalized database system keeps a repository of user profiles (Koutrika, 2003). Information in profiles is either inserted explicitly by the user or collected implicitly by monitoring user interaction with the system (profile creation). When a new query is issued, query personalization proceeds in three stages: (a) preference selection (preferences relevant to the query are extracted from the user profile); (b) preference integration

Figure 2. Personalized database system

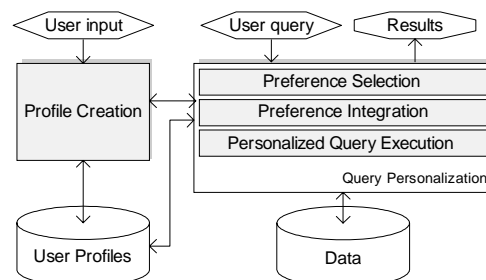


Table 1. A summary of query personalization issues

<p><b>User Preference Modeling</b> Expressing user preferences in queries and storing user preferences in profiles</p> <p><b>Preference Query Languages</b> Languages that allow for the expression of user preferences in queries</p> <p><b>Preference Combination and Ranking</b> Ways in which preferences are combined. Ranking results based on the preferences satisfied</p>	<p><b>Preference Elicitation</b> Algorithms for the (semi-) automatic derivation of user preferences</p> <p><b>Query Personalization Logic</b> Specifying how a query and a profile are combined in order to provide personalized answers</p> <p><b>Query Personalization Algorithms</b> Algorithms for the implementation of query personalization based on a preference model and specific logic</p>
--	--

(these preferences are integrated into the query producing one that will be executed); and (c) personalized query execution (the personalized query is executed and returns results ranked based on their interest). The general architecture of such a system is depicted in Figure 2. Towards personalized database systems, several challenging issues need to be addressed. Table 1 summarizes many of them.

### User Preference Modeling and Preference Query Languages

There is a plethora of preference types: conditional and unconditional, positive and negative, atomic and aggregate, and so forth. Therefore, an effective and efficient preference model has to combine expressivity and concision. That information stored in database systems is structured as opposed to unstructured information stored in information retrieval systems lends itself to specification of more expressive user models than simple keyword-based ones used in the latter systems.

User preferences and profiles have recently attracted a broader interest in the database community. Database research approaches fall into two categories: those dealing with the expression of preferences at the query level and those dealing with the representation and storage of preferences in user profiles.

Database research approaches dealing with the expression of preferences at the query level may be qualitative or quantitative. In the *qualitative* approach, preferences between tuples in the answer to a query are specified directly, typically using binary preference relations. Two frameworks have been independently proposed, in which preference relations are defined using logical formulas (Chomicki, 2003) or special preference constructors (Kießling & Köstler, 2002). Preference relations are embedded into relational query languages through a relational operator that selects from its input the set of the most preferred tuples. This operator is variously called *winnow* (Chomicki, 2003), *BMO* (Kießling & Köstler, 2002a). Kießling and Köstler (2002b)

propose a structured query language, *Preference-SQL* that is an extension of SQL for expression and handling of user preferences. In the *quantitative* approach, preferences are specified indirectly using scoring functions that associate a numeric score with every tuple of the query answer. A framework for using and combining quantitative preferences in queries has been proposed (Agrawal & Wimmers, 2000). Several efforts have focused on algorithms for efficiently answering top-*K* queries (Bruno, Chaudhuri, & Gravano, 2002; Chaudhuri & Gravano, 1999; Hristidis, Koudas, & Papakonstantinou, 2001). On the other hand, models for representing and storing user preferences in profiles have been recently presented (Koutrika & Ioannidis, 2004a, 2004b).

None of the aforementioned proposals encompasses all possible preference notions. The difference in foci, as well as variations in terminology, makes the results obtained in one area difficult to use in another. This situation calls for specialized and collaborative research towards preference models that are appropriate for expression of preferences at the query level and the formulation of profiles in a unifying manner.

### Preference Combination and Ranking

Different preferences may be combined. What is the user interest in a combination of preferences? How are results ranked based upon the user profile? These are issues that must be addressed by a personalized database system. Koutrika and Ioannidis (2004b) propose the use of ranking functions for estimating the degree of interest in results, based on preferences satisfied. Yet, much experimentation is required towards an appropriate and intuitive solution.

### Preference Elicitation

Users may manually create and update their profiles by inserting and editing information related to their preferences. Alternatively, the system may collect and store information in profiles by monitoring user interac-

tion. Both approaches have pros and cons. The first one is easy to implement and allows users to be in charge of what information is stored in their profiles. However, users' reluctance to provide personal information results in sparse or out-of-date user profiles. For this reason, techniques for (semi-)automatic construction of profiles should be employed as well. Nevertheless, users should be able to inspect and modify their personal information.

Holland, Ester, and Kießling (2003) describe preference mining techniques based on the model presented in Kießling and Köstler (2002). However, existing work has primarily focused on the population of simple, keyword profiles used in IR systems. Given the wealth of preference types, development of preference elicitation techniques remains an open and challenging research problem. For example, capturing preferences such as "I don't like director W. Allen" or "I like films without violence" is one of the most difficult and challenging issues.

## Query Personalization Logic

The purpose of query personalization is to focus a search by integrating relevant information stored in a user profile into the initial query. Which information is relevant and how this is integrated into a query are issues depending on the query, the user profile, and various aspects that comprise the *query context*, such as the search goal, the time of the request, the user location, the device of access, and so forth. Consideration of the query context is called *contextualization* (Pitkow et al., 2002). All the above are captured by the personalization logic adopted by the system, which may be represented as a set of criteria and rules. For instance, a rule may specify that the system should return short answers based on a few top user preferences, whenever users access information through a cellular phone. Accurate and effective personalization greatly depends on the personalization logic adopted, thus development of an appropriate and extensible set of rules and criteria is both crucial and challenging. Existing approaches primarily deal with the construction of personalized answers based on the query issued and a user profile (Koutrika, 2003). Consideration of the query context remains an open issue. Detection of a query's context and context switches as well as formulation of system answers based on these, present great research challenges.

## Query Personalization Implementation

Efficient algorithms need to be designed and implemented for implementation of each step of the query personalization process. Naturally, these depend on the user model and query personalization logic used by a personalized system. Such algorithms have been pro-

posed in literature for determining which preferences are syntactically relevant to a query (based on the database structure) and for generating personalized answers (Koutrika & Ioannidis, 2004b). A challenging issue is to determine which preferences are semantically relevant to a query in a given context. For example, a preference for director W. Allen is semantically related to a query about comedies; on the other hand, a preference for director M. Tarkowski is semantically conflicting with the same query. For this purpose, additional knowledge needs to be captured.

## FUTURE TRENDS

Personalized database information access opens the door to a new set of challenges and opportunities for the future.

Preference is a fundamental notion in areas of applied mathematics, philosophy, and computer science that deal with decisions and choice. In mathematical decision theory, preferences (or utilities) are used to model people's economic behavior. In philosophy, they are used to reason about values, desires, and duties. In artificial intelligence, they capture agents' goals. In databases, they are used for the expression of user query criteria and for the formulation of user profiles. There has been so far little interaction between those areas. The difference in foci and terminology variations make the results obtained in one area difficult to use in another. In my opinion, specialized research and collaboration between them will improve both understanding and modelling of preferences.

Another open research area is the manipulation of multiple user profiles belonging to the same person. Up to now, research efforts have primarily focused on representing a user by a single profile. However, one may have different profiles; for example, one may belong to different groups. Combining and reconciling information stored in diverse profiles, and profile hierarchies are just a few of the challenging topics that need to be addressed towards this direction.

Combining personal preferences with other aspects of a query's context that call for query customization, such as time of day, user location, etc is certainly an outstanding research challenge in the near future.

Furthermore, because query personalization alters the search experience, the user interface needs to provide a way to explain what the system is doing to personalize the experience as well as to undo the personalization. Therefore, an interesting research direction is towards design of user interfaces that allows users to control the extent of the personalization, and can help alleviate inaccurate personalization.

Finally, current efforts propose systems built on top of existing database management systems. It is interesting to explore how database technology can be extended, possibly with new operators and methods, in order to support personalized information access from within.

## CONCLUSION

Traditionally, information access has followed a query-based paradigm. The advent of the World Wide Web and hand-held electronic devices generated the need for a new personalized information access paradigm. Different approaches aim to personalization of the overall user experience at different levels: content selection, content presentation, and user interaction. This article has focused on the level of personalized content selection and, in particular, on query personalization in databases. It has addressed the main issues and research problems in the area and presented state of the art research efforts. This is an emerging hot area and there is a plethora of open challenges. Personalization of search is the next frontier toward significantly increasing search efficiency.

## REFERENCES

Agrawal, R., & Wimmers, E. (2000). A framework for expressing and combining preferences. *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, Dallas, Texas (pp. 297-306).

André, E., & Rist, T. (2002). From adaptive hypertext to personalized Web companions. *Communications of the ACM*, 45(5), 43-46.

Belkin, N., & Croft, W. B. (1992). Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12), 29-38.

Bruno, N., Chaudhuri, S., & Gravano, L. (2002). Top-k selection queries over relational databases: Mapping strategies and performance evaluation. *ACM Transactions on Database Systems*, 27(2), 153-187.

Chaudhuri, S., & Gravano, L. (1999). Evaluating top-k selection queries. *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB)*, Edinburgh, Scotland (pp. 397-410).

Chen, J., DeWitt, D., Tian, F., & Wang, Y. (2000). NiagaraCQ: A scalable continuous query system for internet databases. *Proceedings of the ACM International Confer-*

*ence on Management of Data (SIGMOD)*, Dallas, Texas (pp. 379 - 390).

Chomicki, J. (2003). Querying with intrinsic preferences. *ACM Transactions on Database Systems*, 28(4), 1-39.

Foltz, P., & Dumais, S. (1992). Personalized information delivery: An Analysis of information filtering methods. *Communications of the ACM*, 35(12), 51-60.

Glover, E., Lawrence, S., Birmingham, W., & Lee Giles, C. (1999). Architecture of a metasearch engine that supports user information needs. *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, Kansas City, Missouri (pp. 210-216).

Hristidis, V., Koudas, N., & Papakonstantinou, Y. (2001). PREFER: A system for the efficient execution of multi-parametric ranked queries. *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, Santa Barbara, California (pp. 259-270).

Holland, S., Ester, M., & Kießling, W. (2003). Preference mining: A novel approach on mining user preferences for personalized applications. *PKDD, LNAI 2838*, Cavtat-Dubrovnik, Croatia, 204-216.

Karypis, G. (2001). Evaluation of item-based top-n recommendation algorithms. *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, Atlanta, Georgia (pp. 247-254).

Kießling, W., & Köstler, G. (2002a). Foundations of preferences in database systems. *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB)*, Hong Kong, China (pp. 311-322).

Kießling, W., & Köstler, G. (2002b). Preference SQL-design, implementation, experiences. *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB)*, Hong Kong, China (pp. 990-1001).

Koutrika, G. (2003). A personalization framework for database queries. *Proceedings of the 2<sup>nd</sup> Hellenic Data Management Symposium (HDMS)*, Athens, Greece.

Koutrika, G., & Ioannidis, Y. (2004a). Personalization of queries in database systems. *Proceedings of the International Conference on Data Engineering (ICDE)*, Boston (pp. 597-608).

Koutrika, G., & Ioannidis, Y. (2004b). Personalized queries using a generalized preference model. *Proceedings of the 3<sup>rd</sup> Hellenic Data Management Symposium (HDMS)*, Athens, Greece.

Liu, L., Pu, C., & Tang, W. (1999). Continual queries for internet scale event-driven information delivery. *IEEE*



*Transactions on Knowledge and Data Engineering*, 11(4), 610-628.

Liu, F., Yu, C., & Meng, W. (2002). Personalized Web search by mapping user queries to categories. *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, McLean, Virginia (pp. 558-565).

Pitkow, J., Schutze, H., Cass, T., Cooley, R., Turnbull, D., Edmonds, A., et al. (2002). Personalized search. *Communications of the ACM*, 45(9), 50-55.

Sakagami, H., Kamba, T., Sugiura, A., & Koseki, Y. (1997). Learning personal preferences on online newspaper articles from user behaviours. *Proceedings of the 6th International World Wide Web Conference*, Santa Clara, CA (pp. 291-300).

Semeraro, G., Degemmis, M., Lops, P., Thiel, U., & L'Abbate, M. (2003). A personalized information search process based on dialoguing agents and user profiling. *ECIR, Lecture Notes in Computer Science 2633*, 613-621.

Shahabi, C., Banaei-Kashani, F., Chen, Y., & McLeod, D. (2001). Yoda: An accurate and scalable Web-based recommendation system. *Cooperative Information Systems, Lecture Notes in Computer Science 2172*, 418-432.

Smyth, B., Bradley, K., & Rafter, R. (2002). Personalization techniques for online recruitment services. *Communications of the ACM*, 45(5), 39-40.

## KEY TERMS

**Filter-Based Information Access Approaches:** System responses are filtered on the basis of a rudimentary user profile storing long-term user interests.

**Personalization:** The approach of providing an overall customized, individualized user experience by taking into account the needs, preferences and particular characteristics of a user (or group of users).

**Personalized Database System:** A database system that provides personalized answers in response to a user request by dynamically considering relevant user information stored in user profiles. Its basic modules include a query personalization module, and a profile creation module.

**Personalized Information Access Approaches:** Information is returned to the user, taking into account the query issued and particular characteristics of a user.

**Query Personalization:** The process of dynamically enhancing a query with relevant preferences stored in a user profile with the intention of focusing a search and providing individualized answers.

**Query-Based Information Access Approaches:** Information is returned to the user, taking into account only the query issued.

**User Profile:** System-level representation (model) of a user, used for customizing system responses.



# Database Replication Protocols

**Francesc D. Muñoz-Escóí**

*Instituto Tecnológico de Informática, Spain*

**Luis Irún-Briz**

*Instituto Tecnológico de Informática, Spain*

**Hendrik Decker**

*Instituto Tecnológico de Informática, Spain*

## INTRODUCTION

Databases are replicated in order to obtain two complementary features: performance improvement and high availability. Performance can be improved when a database is replicated since each replica can serve read-only accesses without requiring any coordination with the rest of replicas. Thus, when most of the application accesses to the data are read-only, they can be served locally without preventing other processes to access the same or other replicas. Moreover, a careful coordination management can ensure that the failure of one or more replicas does not compromise the availability of the database as long as at least one of the replicas is alive.

## BACKGROUND

Initially, database replication management had been decomposed into two tasks: concurrency control and replica control, usually tackled by different protocols. In the first case, solutions known from the non-replicated domain have been evolved into distributed concurrency control protocols (Bernstein & Goodman, 1981), based either on the two-phase-locking (2PL) or some timestamp-ordering protocol. In the second case, replica control management was based on voting techniques (Gifford, 1979). These voting techniques assign a given number of votes to each replica, usually one, requiring that each read access collects a read quorum (“r” votes) and each write access a write quorum (“w” votes). The database must assign version numbers to the records being replicated so that sum of the values of “r” and “w” is ensured to be greater than the total number of votes and that “w” is greater than half the amount of votes. Thus, it can be guaranteed that each access to the

data reaches at least one copy of the latest version number for each record. This approach ensured consistency, but the resulting communication costs could be prohibitively high.

Voting replica-control protocols with improved features were still used in the next decade, also including routines for system partition management using dynamic voting approaches (Jajodia & Mutchler, 1990).

However, replication management is not easily achieved when concurrency and replica control are merged, since what the replica control protocols do for ensuring consistency has to be accepted by the used concurrency control. Unfortunately, deadlocks and transaction abortions are quite common when both protocols are merged. Thus, it seems adequate to find better solutions for this global management task, that is, replication protocols that cater for both concurrency and replica controls. A first example of this combined technique is Thomas (1979), where a special kind of voting algorithm is combined with timestamp-based concurrency control. However, his solution still relies on simple communication primitives and is not efficient enough, both in terms of response time and abortion rate. In fact, efficient reliable or atomic broadcast protocols were not produced until the mid-1980s (Birman & Joseph, 1987) and could not yet be used in these first stages.

Thus, new replication techniques were introduced in the database arena, having evolved from the process replication approaches known from distributed systems. Depending on the used criteria, several classifications are possible. However, it is useful to distinguish, in a first step, between lazy and eager techniques (Gray, Helland, O’Neil & Shasha, 1996) and limit attention to considering only the update propagation strategy. Both of these approaches are described hereafter, where additional criteria are considered in order to refine this taxonomy.

## EAGER REPLICATION

Eager replication propagates the updates of a transaction before the transaction commits. This ensures the consistency of all replicas, but all of the computing time needed for communication then needs to be added to the transaction lifetime, thus causing response times that are longer than those of lazy replication techniques. However, if an atomic broadcast protocol (Hadzilacos & Toueg, 1993) is used, concurrency control can be dealt with locally, so the overall communication costs can be kept low.

Several eager replication techniques exist. According to Wiesmann et al. (2000), the following three characteristics are used to classify the following techniques.

### Server Architecture (Gray et al., 1996)

Considers where updates of a given data item are initially processed. There are two options:

- **Update Everywhere (UE):** The updates can be initially done in any of the replicas of the data item.
- **Primary Copy (PC):** For each data item, a distinguished replica exists, its primary copy. Only the primary copy may initially process an update of such a data item; that is, there is only one active replica, and it is always the same one.

### Server Interaction

Analyzes how many messages are exchanged by the servers during transaction processing. There are two alternatives:

- **Linear Interaction (LI):** Servers exchange messages for each operation involved in the transaction. As a result, the number of messages depends on the transaction length.
- **Constant Interaction (CI):** Servers exchange a fixed number of messages. The typical case is one update message at the end of the transaction.

### Transaction Termination

Considers the actions needed by the replicas to decide the result of a transaction. This depends on the determinism of the algorithm. Again, two options exist:

- **Voting Termination (VT):** An additional round of messages is needed to decide whether the trans-

action has to be committed or aborted. The traditional two-phase commit protocol is a typical example of this kind of termination.

- **Non-Voting Termination (NT):** This option is applicable when all replicas may decide locally on the completion of a transaction. To this end, a deterministic and symmetrical distributed algorithm is needed, in general. However, determinism is not needed when transactions do not conflict; that is, two or more transactions accessing disjoint sets of records may be executed or terminated in any order in different replicas.

Combining the three characteristics above yields eight classes of replication protocols:

1. UE-CI-VT: “Update everywhere” propagation, with “constant interaction” and “voting termination”.
2. UE-CI-NT: “Update everywhere” propagation, with “constant interaction” and “non-voting termination”.
3. UE-LI-VT: “Update everywhere” propagation, with “linear interaction” and “voting termination”.
4. UE-LI-NT: “Update everywhere” propagation, with “linear interaction” and “non-voting termination”.
5. PC-CI-VT: “Primary copy” update propagation, with “constant interaction” and “voting termination”.
6. PC-CI-NT: “Primary copy” update propagation, with “constant interaction” and “non-voting termination”.
7. PC-LI-VT: “Primary copy” update propagation, with “linear interaction” and “voting termination”.
8. PC-LI-NT: “Primary copy” update propagation, with “linear interaction” and “non-voting termination”.

For reducing the communication costs of an eager protocol, CI is better than LI. Traditionally, eager protocols mostly used the LI-VT combination, either with UE or with PC. The authors of this classification proposed UE-CI-VT and UE-CI-NT as the best possible alternatives for eager protocols; some examples of them have been presented in Kemme and Alonso (2000). Each of these protocols needs atomic broadcast. The protocol described in Irún, Muñoz, Decker, and Bernabéu (2003) belongs to the UE-CI-VT class and only requires uniform reliable broadcast, which is faster than atomic broadcast. However, since reliable broadcast does not guarantee that all updates are delivered in the same order in all replicas, a voting termination procedure is needed, since each replica cannot determine locally if a transaction has to be committed or aborted. Thus, using reliable broadcast without voting, the UE-CI-NT technique cannot be implemented.

The UE propagation technique is less scalable than the PC approach, since a coordination phase is needed to find out if different transactions collide, while in the PC

alternative, each “primary copy” replica is able to find and manage the transaction conflicts (Gray et al., 1996). In Patiño, Jiménez, Kemme, and Alonso (2000), two UE-CI-NT protocols are described. These protocols use a concurrency control scheme based on “conflict classes”, similar to locks but easier to compute, and requiring only local processing for these control tasks. Moreover, both protocols use an atomic broadcast protocol with optimistic delivery that is able to reduce the communication costs, needing the same number of message rounds as a reliable broadcast. Consequently, these algorithms are easily scalable, thus, eliminating one of the main disadvantages of UE eager protocols.

## LAZY REPLICATION

Lazy replication delays the propagation of updates until the transaction has committed. Once committed, updates are propagated. This approach enables a fast transaction completion but does not always ensure replica consistency and may therefore lead to a high abortion rate. Despite its disadvantages, this technique has been used in several commercial DBMSs. Moreover, it is the only possible option for replicating mobile or disconnected databases.

Depending on the server architecture, as with eager protocols, two classes of lazy protocols can be distinguished: update everywhere (UE) and primary copy (PC).

Update everywhere protocols allow that any one of the replicas may update its local data directly, while transmitting the updates to the other replicas thereafter. If the concurrency control tasks are checked before commit time, this may lead to a high abortion rate. Otherwise, a reconciliation procedure is needed to merge the updates of conflicting transactions. For concurrency control purposes, a timestamp-based solution is commonly used. However, it is extremely difficult, if not impossible, to provide one-copy serializability guarantees for these kinds of protocols. Several algorithms providing one-copy serializability have been published, but they either use an additional precommit phase that may lead later to the abortion of the transaction (Agrawal, El Abbadi & Steinke, 1997), violating thus a strict lazy replication definition; or use a specific broadcasting topology which prevents that some of the replicas may initiate update broadcasts, thus resulting in something similar to the primary copy approach in eager systems, while limiting the general usefulness of such solutions (Anderson, Breitbart, Korth & Wool, 1998).

Since in primary copy protocols the first access is always managed by the primary replica, the latter may use locks or timestamps to avoid or detect conflicts between transactions. If locks are used, deadlocks may arise, and

additional deadlock-managing protocols are then needed. When timestamps are used, the resulting abortion rate will be high, at least when compared to eager solutions. Primary copy solutions have been used in some commercial database systems. For instance, they are still supported in Sybase Replication Server (Sybase, 2003). In these systems, two opposed trends can be identified. The first one uses replication to ensure only availability, but not to improve performance. In that case, the replicas behave as standby copies of the primary replica. In the second one, replication is mainly used to enhance performance, and serializable consistency is not maintained. However, in this context, it is worth noting that most applications can be run perfectly with relaxed consistency modes or isolation levels. Indeed, the default isolation level of most relational DBMSs is not serializable, but read committed (for instance, in PostgreSQL) or repeatable read.

As observed above, lazy protocols are the unique option for mobile or disconnected database. One of the first studies in this area is Gray et al. (1996). It describes a two-tier protocol that can manage mobile replicas. To this end, Gray et al. classify the replicas into two distinct groups:

- **Base nodes:** Those that are always interconnected. They use an eager replication protocol to propagate the updates among them.
- **Mobile nodes:** Those that are usually disconnected. They propose tentative update transactions to data items owned by base nodes.

This protocol requires that mobile nodes maintain two data item versions: a local one, and a so-called best-known master version. Thus, when a mobile node connects to a base node, it proposes tentative update transactions to a “primary copy” base node. These transactions are re-executed in the base node; they may succeed or not. Moreover, tentative transactions are designed as commutable with other transactions, improving thus their probability of successful termination. If a tentative transaction is rejected, its mobile node has to reconcile its updates. Additionally, the connection procedure updates the mobile replica, applying the updates it has missed during the disconnected period.

Regular updates can be transformed into commutable updates if they do not overwrite data items with new values, but only increment or decrement the data item’s previous value. This is the principle used in the mobile protocol described above. Similar solutions were used in the second protocol of Patiño et al. (2000), as mentioned in the section on eager protocols above. The aim of both protocols is the same: to reduce

the abortion rate. This problem is particularly important in all mobile environments.

## FUTURE TRENDS

Future trends in database replication are, among others, the improvement of mobile databases support, the development of hybrid replication protocols, and the minimization of the blocking periods during replica recoveries. They are outlined in the following paragraphs.

Currently, only a few replication protocols for mobile databases exist; it is an open research problem where lazy replication protocols will be applied. Protocols designed for replication and consistency maintenance in mobile databases are easily portable to mere file systems, thus, serving as a basis for constructing tools to ensure the consistency of file system replicas in portable computers and PDAs, for instance.

Hybrid protocols are a third kind of update propagation solution when database replication is considered. The COLU protocol (Irún, Muñoz & Bernabéu, 2003) is an example of such a hybrid solution. By default, it is an UE-CI-VT lazy protocol, but it allows the configuration of the number of replicas receiving updates at commit time (i.e., the number of synchronous replicas), before the transaction is terminated. So, it may behave as a pure lazy protocol when no synchronous replicas are used, or as a pure eager protocol when all replicas are configured as synchronous. Moreover, intermediate configurations are also possible, guaranteeing that transactions are not lost when failures arise. This hybrid protocol is able to use a lazy recovery strategy, minimizing the recovery time.

Database replication protocols always need a recovery protocol, but traditionally the research works in this area did not focus on such recovery algorithms. This situation has recently changed. Kemme, Bartoli, and Babaoglu (2001) proposed a multiple-stage lazy recovery that minimized the blocking time, both in the recovering and source replicas. Jiménez, Patiño, and Alonso (2002) improved this solution, distributing the source role among several active replicas and reducing the amount of logged data to be transmitted, using periodic checkpoints. Further work remains to be done in this domain in order to reduce the amount of transmitted data or the blocking time, minimizing thus the recovery delays.

## CONCLUSION

Database replication has commonly used lazy protocols in commercial DBMSs, thus, ensuring good perfor-

mance but without providing one-copy serializability. To overcome resulting problems, eager replication protocols with group communication support have been proposed. The use of atomic broadcast protocols has enabled the development of “update everywhere” propagation with “constant interaction” and “non-voting termination” eager protocols. Such solutions, combined with commutable transactions, ensure one-copy serializability with a performance similar to lazy protocols, plus easy scalability and very low abortion rates. Although these solutions are not available in commercial database management systems yet, we expect them to have a marketable impact soon.

New areas of research in the database replication arena are those of mobile databases and hybrid replication. Mobile databases require a lazy replication solution. This ensures that lazy protocols will again be of interest for database researchers and developers. The main problem to be solved in this area relates to transaction reconciliation procedures. No fully automated procedure yet exists.

Hybrid replication protocols will be able to provide the appealing properties of the new-generation eager protocols when one-copy serializability is needed, and the very good performance of traditional lazy protocols when a more relaxed consistency model is feasible.

## REFERENCES

- Agrawal, D., El Abbadi, A., & Steinke, R. (1997). Epidemic algorithms in replicated databases. *Proceedings of the 16th ACM Symposium on Principles of Database Systems* (pp. 161-172).
- Anderson, T., Breitbart, Y., Korth, H.F., & Wool, A. (1998). Replication, consistency, and practicality: Are these mutually exclusive? *Proceedings of the ACM SIGMOD International Conference on the Management of Data* (pp. 173-182).
- Bernstein, P., & Goodman, N. (1981). Concurrency control for distributed database systems. *ACM Computing Surveys*, 13(2), 185-221.
- Birman, K.P., & Joseph, T.A. (1987). Reliable communication in the presence of failures. *ACM Transactions on Computer Systems*, 5(1), 47-76.
- Gifford, D.K. (1979). Weighted voting for replicated data. *Proceedings of the 7th ACM Symposium on Operating System Principles* (pp. 150-162).
- Gray, J., Helland, P., O’Neil, P., & Shasha, D. (1996). The dangers of replication and a solution. *Proceedings of the ACM SIGMOD Conference* (pp. 173-182).



Hadzilacos, V., & Toueg, S. (1993). Fault-tolerant broadcasts and related problems. In S. Mullender (Ed.), *Distributed systems* (pp. 97-145). Reading MA: Addison-Wesley.

Irún, L., Muñoz, F.D., & Bernabéu, J. (2003). An improved optimistic and fault-tolerant replication protocol. *Lecture Notes in Computer Science*, 2822, 188-200.

Irún, L., Muñoz, F.D., Decker, H., & Bernabéu, J. (2003). COPLA: A platform for eager and lazy replication in networked databases. *Proceedings of the 5th International Conference on Enterprise Information Systems* (pp. 273-278).

Jajodia, S., & Mutchler, D. (1990). Dynamic voting algorithms for maintaining the consistency of a replicated database. *ACM Transactions on Database Systems*, 15(2), 230-280.

Jiménez, R., Patiño, M., & Alonso, G. (2002). Non intrusive, parallel recovery of replicated data. *Proceedings of the 21st IEEE Symposium on Reliable Distributed Systems* (pp. 150-159).

Kemme, B., Bartoli, A., & Babaoglu, O. (2001). Online reconfiguration in replicated databases based on group communication. *Proceedings of the International Conference on Dependable Systems and Networks* (pp. 117-130).

Kemme, B., & Alonso, G. (2000). A new approach to developing and implementing eager database replication protocols. *ACM Transactions on Database Systems*, 25(3), 333-379.

Patiño, M., Jiménez, R., Kemme, B., & Alonso, G. (2000). Scalable replication in database clusters. *Lecture Notes in Computer Science*, 1914, 315-329.

Sybase, Inc. (2003). Replication strategies: Data migration, distribution and synchronization (Technical White Paper). Retrieved January 23, 2005, from <http://www.sybase.com/detail/1,6904,1028711,00.html>

Thomas, R.H. (1979). A majority consensus approach to concurrency control for multiple copy databases. *ACM Transactions on Database Systems*, 4(2), 180-209.

Wiesmann, M., Pedone, F., Schiper, A., Kemme, B., & Alonso, G. (2000). Database replication techniques: A three-parameter classification. *Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems* (pp. 206-215).

## KEY TERMS

**Active Replica:** The replica that directly processes a given transaction, generating the updates that later will be transmitted to the other, passive replicas. The active status of a replica functionally depends on the given transaction.

**Asynchronous Replication:** Lazy replication; that is, all updates of a transaction (if any) are transmitted to passive replicas once the transaction is committed but never ahead of commit time.

**Atomic Broadcast:** Requires that each correct process delivers all messages in the same order, that is, a reliable broadcast with total order.

**Eager Replication:** See Synchronous Replication

**Hybrid Replication:** A replication technique using protocols that may either behave as eager or as lazy, depending on the given system configuration.

**Lazy Replication:** See Asynchronous Replication

**Passive Replica:** A replica that does not directly process the transaction. Instead, it only applies updates received from active replicas.

**Reconciliation Procedure:** When, in lazy protocols, two conflicting transactions have been committed before conflict detection, this procedure is needed to adequately reconcile and merge the respective updates.

**Reliable Broadcast:** Requires that each correct process delivers the same set of messages and that the set includes each message broadcast by correct processes, but no spurious messages.

**Synchronous Replication:** Eager replication; that is, transaction updates are propagated before the transaction is committed.



# Database Support for Workflow Management Systems

Francisco A. C. Pinheiro

Universidade de Brasília, Brazil

## INTRODUCTION: WORKFLOW SYSTEMS

A workflow is a series of work processes performed under rules that reflect the formal structure of the organization in which they are carried out and the relationships between their various parts. Workflow applications are software applications used to automate part of workflow processes. They run under the control of a workflow management system (WfMS). The WfMS usually comprises an organizational model, describing the process structure, and a process model, describing the process logic. The Workflow Management Coalition (WfMC, 2004) publishes a set of workflow definitions and related material, including a reference model.

Databases are commonly used as a WfMS supporting technology. Not only workflow data are maintained in databases, but also the rules governing processes can be stored in database schemas. Database functionality can be used both for defining and managing process models as well as for environment notification and process enactment. This article shows how particular database-related technologies can be used to support WfMS.

Table 1 relates workflow issues and the database technologies that can be used to deal with them. It

summarizes the content of this article presenting the relationships discussed in the text. The next two sections discussing workflow management issues and database support are related to the columns and lines of the table and provide an explanation for these relationships. Only the general relationships stressed in the text are shown. For example, data replication and partitioning have an influence on scalability and privacy, but the table does not show the same influence with respect to distributed database technology, although data replication and partitioning are basic strategies of distributed databases.

## BACKGROUND: WORKFLOW MANAGEMENT ISSUES

Workflow applications are usually complex, distributed applications consisting of activities performed by people playing different roles, sometimes involving multiple departments and organizations. It is not unusual for different workflows to be governed by different WfMS running under a variety of platforms, application servers, and communications middleware. This situation requires ways of dealing with diversity and complexity

Table 1. General relationships between workflow issues and database technologies

		ENVIRONMENT			PEOPLE			PROCESS			DATA		
		Diversity	Interoperability	Scalability	Collaborative work	Flexibility	Evolution	Assignment of tasks	Consistency	Changes	Privacy	Location	Semantic heterogeneity
DATABASE TYPES	Distributed	✓	✓										✓
	Parallel			✓									
	Multi	✓	✓										
	Active							✓				✓	
DATABASE TECHNOLOGIES	Data replication			✓							✓		
	Data partitioning			✓						✓			
	Synchronisation techniques							✓	✓				
	Transaction models				✓	✓			✓				
	Schema					✓	✓		✓	✓			✓
	Metadata				✓	✓	✓		✓		✓	✓	

and imposes a need for interoperability and scalability. Other issues like heterogeneity, consistency, privacy, and flexibility naturally arise when we think about the configuration of such an environment and the development of workflow applications. Stohr and Zhao (2001) present a detailed discussion of workflow issues.

### Environment issues

Workflow diversity ranges from organization to infrastructure. Organizational diversity involves different processes, rules, and ways of organizing work. Infrastructure diversity involves different platforms, communication protocols, programming languages, and data formats. Interoperability is needed for applications running in such a diverse environment. The changing environment also makes scalability crucial to cope with the addition of new organizational units and redistribution of work to other (sometimes geographically distant) players. Support to these issues may be found in technologies incorporated into distributed, parallel, and multi-databases.

### People Issues

Collaborative work should not be hindered by overly rigid procedures. There should be room to modify processes as necessary, making possible to follow different paths by changing the ordering of activities or suppressing some of them. Flexibility and evolution are necessary to cope with different views about the organization of work and use of resources. The need for flexible and evolving environments is supported by metadata descriptions and schema evolution. Advanced transaction models should be in place to reconcile flexibility with consistency.

### Process Issues

A number of workflow patterns for relevant process issues is described by van der Aalst et al. (2003). Particularly, the assignment of tasks is a highly dynamic activity that should be performed promptly and correctly. Dynamic team formation (Georgakopoulos, 2004), with members being added and released as a process goes on, requires the system to be aware of its environment. Active databases provide some sort of environment awareness and synchronization techniques may be used to keep processes and their data in a consistent state.

### Data Issues

Data change in different ways, including location, as a result of people performing their activities. To locate the relevant data in a changing environment, with autonomous collaborating agents, is a difficult task. A complicating factor in distributed environments is that of semantic heterogeneity, in which people assign different meanings to the same or closely related data. We also have the issue of transparency in which some data should be shared while others should be kept private.

The use of metadata description and schema evolution help to maintain data availability and consistency, and to deal with semantic heterogeneity. Techniques of data replication and partitioning have an impact on privacy, and active database technology may be employed to notify and deliver the needed data to agents.

## DATABASE SUPPORT

This section relates particular database technologies to the workflow issues discussed above.

### Distributed, Active, and Multi-Databases

Distributed databases are used to manage data stored in several places, while maintaining a uniform control over the operations that use and change them. Parallel databases allow the parallel execution of actions and multi-databases, also referred to as federated or heterogeneous databases, make possible the coordinated use of several databases. These types of databases are already being used by workflow management systems that apply their internal mechanisms to deal with diversity, scalability, interoperability, and heterogeneity.

Active database management systems make possible the system to be aware of what is happening around it and to react properly and spontaneously, for example, firing triggers and alerters. They are useful for the assignment of tasks and location of data and provide an appropriate support to build adaptation mechanisms into workflow systems (Bernstein et al., 1998).

### Data Replication and Partitioning

Data may be partitioned or replicated in different locations to improve performance and scalability and to assure privacy. Data replication and partitioning are basic strategies used in distributed and parallel databases. An optimal strategy to set the right combination of replication

and partitioning depends on the application, but it is possible to determine general data access patterns in case of WfMS (Schuster & Heinel, 1997).

## **Metadata Description and Database Schemas**

Metadata descriptions are employed by database systems to make them aware of other database needs (Bernstein et al., 1998) and also to locate relevant data in distributed systems, providing enough background information to judge the suitability and trustworthiness of the data sources. It is an important technology concerning collaboration, flexibility, and evolution in workflow systems.

Database schemas reflect the rules governing processes and their data. Schema evolution (Lerner, 2000) gives the user the ability to dynamically change the schema of a data source. This is important to preserve flexibility and guarantee consistency in changing environments.

## **Transaction Models**

New transaction models, allowing flexible ways of dealing with consistency and recovery is paramount to support collaborative work. Nested transactions and open nested transactions are well-known advanced transaction models, allowing composite units of work to be reflected as transaction structures. A survey of advanced transactions models can be found in Barghouti and Kaiser (1991). Some workflow applications may require the definition of application-specific transaction models (Geogakopoulos, Hornick & Manola, 1996). Multiuser transaction for cooperative applications is described in Wiczerzycki (1999).

## **Database Synchronization Techniques**

Concurrency requires synchronization to guarantee the integrity of concurrent data and processes. A number of synchronization techniques, such as fetch-and-add, set-and-test, and wait-free synchronization, are already being used by database systems (Herlihy, 1991) and may be applied by WfMS.

## **FUTURE TRENDS**

There should be advances in all areas related to the flexible, cooperative, and diversified ways of doing business. This points to an increasing use of multiple WfMS engines to support distributed workflow process execution. We should expect more research in the areas of flexible coordination, awareness provision, team scalability, and integration of workflow systems with

computer support for cooperative work (CSCW) and content management technologies (Geogakopoulos, 2004).

The overall trend is to increase flexibility and to incorporate controlled inconsistency, allowing it to exist for a while and be solved later in an application by application basis (Dourish, 1998). For example, we can have synchronization methods designed to minimize divergence between source data and cached copies, instead of trying to maintain exact synchronization (Olston & Widom, 2002). Customized transaction models (Geogakopoulos, 2004), advances on metadata description (Roantree, 2002), and schema evolution (Pittas, Jones & Gray, 2001) should all have a positive impact on collaborative work.

## **CONCLUSION**

A WfMS implemented on top of a database system is a special database application. Therefore, it helps to understand how advances on databases as a supporting technology may be applied to build more useful workflow applications and, on the other hand, how workflow application needs may drive the improvement of database technologies.

Nevertheless, workflow applications are highly human-centered and should not be limited by technology. Organizational requirements have to be taken into consideration when developing workflow applications and in choosing an appropriate WfMS (Silva & Pinheiro, 2003). These requirements may impose restrictions that will never be completely solved by any supporting technology. For example, the most challenging kind of diversity is cultural diversity. It may be ameliorated using techniques to deal with semantic heterogeneity, but there is more to culture than differences in the interpretation of terms.

## **REFERENCES**

- Barghouti, N., & Kaiser, G. (1991). Concurrency control in advanced database applications. *ACM Computing Surveys*, 23(3), 269-317.
- Bernstein, P., Brodie, M., Ceri, S., DeWitt, D., Franklin, M., Garcia-Molina, H., et al. (1998). The asilomar report on database research. *SIGMOD Record*, 27(4), 74-80.
- Dourish, P. (1998). Using metalevel techniques in a flexible toolkit for CSCW applications. *ACM Transactions on Computer-Human Interaction*, 5(2), 109-155.

Georgakopoulos, D. (2004). Teamware: An evaluation of key technologies and open problems. *Distributed and Parallel Databases*, 15(1), 9-44.

Georgakopoulos, D., Hornick, M., & Manola, F. (1996). Customizing transaction models and mechanisms in a programmable environment supporting reliable workflow automation. *IEEE Transactions on Data and Knowledge Engineering*, 8(4), 630-649.

Herlihy, M.P. (1991). Wait-free synchronization. *ACM Transactions on Programming Languages and Systems*, 13(1), 124-149.

Lerner, B.S. (2000). A model for compound type changes encountered in schema evolution. *ACM Transactions on Database Systems*, 25(1), 83-127.

Olston, C., & Widom, J. (2002). Best-effort cache synchronization with source cooperation. *Proceedings ACM International Conference on Management of Data* (pp. 73-84).

Pittas, N., Jones, A.C., & Gray, W.A. (2001). Evolution support in large-scale interoperable systems: A metadata driven approach. *Proceedings of the 12th Australasian Conference on Database Technologies* (pp. 161-168).

Roantree, M. (2002). Metadata management in federated multimedia systems. *Proceedings of the 13th Australasian Conference on Database Technologies* (Vol. 5, pp. 147-155).

Schuster, H., & Heintz, P. (1997). A workflow data distribution strategy for scalable workflow management systems. *ACM Symposium on Applied Computing*, 174-176.

Silva, L.P., & Pinheiro, F.A.C. (2003). Eliciting requirements for identifying workflow categories. *Proceedings of the VI Workshop on Requirements Engineering (WER'03)* (pp. 16-31).

Stohr, E.A., & Zhao, J.L. (2001). Workflow automation: Overview and research issues. *Information Systems Frontiers*, 3(3), 281-296.

van der Aalst, W., ter Hofstede, A.H.M., Kiepuszewski, B., & Barros, A.P. (2003). Workflow patterns. *Distributed and Parallel Databases*, 14, 5-51.

WfMC. (2004). Workflow management coalition. Retrieved April 27, 2005, from <http://www.wfmc.org>

Wieczarzycki, W. (1999). Database model for Web-based cooperative applications. *Proceedings of the 8th Conference on Information and Knowledge Management* (pp. 131-138).

## KEY TERMS

The definitions below are based on Georgakopoulos (2004), Schuster and Heintz (1997), Stohr and Zhao (2001), and WfMC (2004).

**Content Management Systems:** Provide tools for organizing, delivering, and sharing documents and images. Usually used in conjunction with CSCW systems or workflow systems.

**CSCW:** Computer-Supported Cooperative Work. Provides tools for supporting people working together, for example, video and audio conferences, group calendar, e-mail, and text chat. Differs from workflow applications in having more flexibility and less coordination.

**Data Partitioning:** A storage technique through which each data item is assigned to exactly one node. Data operations accessing different data partitions can be executed in parallel. However, if one operation needs to access more than one data partition, the execution is more complicated.

**Data Replication:** A storage technique through which some nodes have copies of the same data. Replication of data is a common method to improve read performance but is rather problematic if data is often updated.

**Metadata Description:** Abstract data description used to describe concrete data items. Usually contains descriptions of data structures, relationships to other data items, location, scope, and other features of the data item.

**Process Enactment:** A WfMS is said to enact the real-world process for each process instance.

**Task Assignment:** The assignment of a task to an agent responsible for it, made by the WfMS. The agent may be a person or a process. When it is a person, the assignment usually implies a notification to the agent; when it is a process, the assignment usually results in its execution.

**Workflow Instance:** Also, process instance. Each execution instance of a process managed by the WfMS.

# Databases for Mobile Applications

## **Indranil Bose**

*University of Hong Kong, Hong Kong*

## **Wang Ping**

*University of Hong Kong, Hong Kong*

## **Mok Wai Shan**

*University of Hong Kong, Hong Kong*

## **Wong Ka Shing**

*University of Hong Kong, Hong Kong*

## **Yip Yee Shing**

*University of Hong Kong, Hong Kong*

## **Chan Lit Tin**

*University of Hong Kong, Hong Kong*

## **Shiu Ka Wai**

*University of Hong Kong, Hong Kong*

## INTRODUCTION

Owing to the rapid development of mobile technology over the past few decades, there have been many different kinds of mobile devices emerging in the market, and most of them work with databases seamlessly. Mobile phone gaming, downloading of ringtones, and e-calendar are some of the prominent examples of mobile applications that require the close integration of mobile devices with databases. Mobile devices take various forms and configurations. The packaging, form factors, hardware platforms, operating system support, and functional capabilities vary across these devices. There are, however, many common attributes shared by the devices, such as notebook computers, pen-based computers, handheld computers, and the like, all of which are used in mobile computing. These devices can be categorized into the following categories according to their functionalities and features, as detailed in Dhawan (1997). They are:

- notebook computers
- personal digital assistants
- tablet computers
- hybrid mobile devices
- mobile phones

In this article, we focus on personal digital assistants (PDA) and mobile phones as they are the most popular and commonly used mobile devices in the industry.

## Mobile Computing Applications

Mobile applications include basic applications like datebook, address book, to-do list, and memos and also horizontal and vertical industry applications that mainly fall into the following three categories (Dhawan, 1997):

- Shrink-wrapped horizontal industry mobile computing applications that can be used in broad segments of various industries, e.g., electronic mail, electronic messaging via paging, and sales force automation.
- Generic horizontal industry applications requiring extensive customization, and these include database access from an information server, computer-aided dispatch (CAD), and intrasite and intersite mobility applications among others.
- Vertical industry applications include the applications that are specific to industries like insurance, banking, airlines, government, utilities, and transportation, e.g., finance industry insurance and financial planning, and stock trading.



The diverse variety of the types of mobile applications demonstrates the reach of mobile computing into almost every facet of personal and business life. One of the applications that is gaining popularity is mobile e-commerce. Mobile e-commerce refers to commercial activities performed electronically. An example of this is an online shopping mall (via the mobile devices to the Internet). Mobile commerce is one of the most popular applications these days in addition to obtaining stock quotes, directions, weather forecasts, and airline flight schedules from mobile devices (Munusamy & Hiew, 2004).

### Comparison Between Mobile Devices and Desktop Computers

Compared to desktop computers, mobile devices have small memory, low computing capabilities, limited interaction facilities, and limited display and network processing capabilities. With recent technological advancements, hybrid devices combining the functionality of mobile phones together with PDAs have been developed. The differences are mainly attributed to their hardware design and system configurations. Table 1 compares desktop computers, PDAs, and mobile phones with respect to their processing power, memory, storage capacity, connection speed, and display. The data presented is current as of June 22, 2004. Data related to specifications for the desktop, PDA, and mobile phone have been downloaded from the Web sites of Dell Inc. and Nokia Inc. and relate to the Dell Dimension 8400 Desktop, the Dell Axim X3 Pocket PC 400 MHz WiFi, and the Nokia 7610, respectively.

From Table 1, it can be observed that mobile devices have smaller memory size and storage capacity as well as display size than desktop computers. So, the amount of data that can be transferred and displayed at a time is less than that of desktop computers. Furthermore, the

Table 1. Comparison between a desktop computer and mobile devices

	Desktop	PDA	Mobile Phone
<b>Processing Power</b>	2.8–3.4 GHz	400 MHz	Unknown
<b>Memory</b>	512 MB–2 MB	64–1024 MB	8 MB (internal)
<b>Storage Capacity</b>	80–400 GB	N/A	N/A
<b>Connection Speed</b>	56 Kbps–100 Mbps	56 Kbps–11 Mbps	Up to 40.3 Kbps
<b>Display</b>	15–19 inch	3.5 inch	1.3 inch

processing power of mobile devices is usually limited when compared with desktop computers. The amount of data that can be processed at a given time is also small. Also, mobile devices have lower connection speeds and less stable network connections. They must have ways to overcome these deficiencies in order to ensure good performance in retrieving data from remote databases.

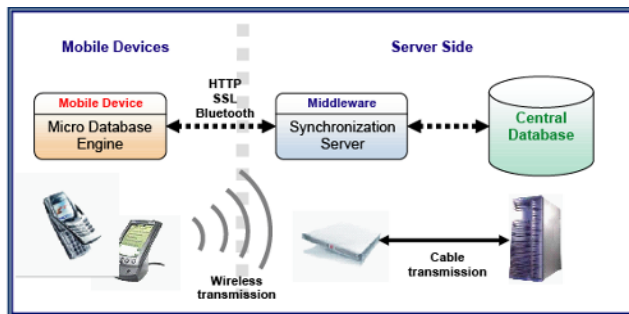
### Challenges for Mobile Devices

Some of the challenges faced by mobile devices when connecting to remote databases include challenges in network connectivity, data transmission, security, and data consistency.

- **Network connectivity:** Mobile devices usually work in an unstable network environment. The network stability is affected by many factors, such as weak signal and strong interference. Without physical network connections, mobile devices often lose connection with the network.
- **Data transmission:** Wireless networks have limited bandwidth compared to traditional cable networks. The slow transmission speed imposes problems in uploading and downloading data. Large network latency constraints also result in long response time.
- **Security:** Any message between the database system and mobile devices is sent over the air, and it is possible for hackers to sniff the message and perform eavesdropping. Advanced encryption and user authentication technology is needed to prevent any such types of hacking activities.
- **Data consistency:** Database applications apply extensive caching and replication to boost performance, which can lead to possible data inconsistency. Mobile devices with little memory storage and slow connection speed cannot obtain all the information from the central database system instantly. The narrow bandwidth of the devices also affects immediate updates from the mobile devices to the database server. It is thus quite difficult to keep data consistent between mobile devices and the database server.

The objective of this article is to give a brief overview on the design of databases for mobile applications and to describe how the database design is currently being done for a successful mobile application called mBroker that is operational in Hong Kong. The article provides a description of the functionalities of the mBroker system and highlights the database design being used by the mBroker solution at the present time.

Figure 1. A typical database architecture for mobile applications



## BACKGROUND

The database design for mobile applications is different from normal database applications running on personal computers. Due to the limited hardware configurations and network settings of mobile devices, database vendors usually provide special database systems and APIs (application programming interfaces) tailored for mobile devices.

Database manufacturers like Oracle, IBM, and Sybase usually follow a similar architecture to build mobile database applications. The main components include a micro database engine, synchronization middleware, and wireless networking. The difference is often the naming of different components. Figure 1 depicts the typical database architecture for mobile applications.

### Micro Database Engine

Mobile devices have limited hardware in terms of processing power, memory size, and battery life. A normal database engine requires a minimum 20 MB memory to operate, which is unavailable for mobile devices. Therefore, database vendors develop robust micro database engines for most mobile devices.

Micro database engines only focus on the most frequently used functionalities which are relevant to mobile applications. These include basic SQL statements, Join, Group By, Order By, scrollable cursors, and simple primary key and foreign key operations. For high-end mobile devices, advanced indexing features are sometimes included to improve performance. Database operations which are rare and less useful are removed from the micro engine. For example, view creation, subqueries, stored procedures, triggers, and user-defined functions are not provided in micro database engines. The DB2

Everyplace and Oracle Lite (Viellard, 2001) are examples of micro database engines.

### Synchronization Middleware

The network capabilities of mobile devices are limited. Real-time online database access requires a large and stable network bandwidth, which is expensive and not always required for mobile applications. Thus, database vendors employ extensive caching and synchronization techniques in mobile database applications. Caching is to preload data to the devices for offline browsing. Users can then browse data seamlessly even if the wireless network is unstable or even unavailable. If instant update to the central server is not necessary, data modification operations are further optimized. Updates are made to the data on the devices first, and the modifications are not reflected on the server database immediately. Data are updated to the remote central database later by synchronization.

Synchronization is done by middleware which is sometimes called a synchronization server or mobile server. Advanced synchronization methods keep data in both mobile devices and the central database consistent. The middleware acts as a middleman between the mobile devices and the central database. It collects changes in the mobile devices and executes appropriate SQL statements to update the central database. At the same time, the middleware also propagates updates in the central database to the mobile devices.

### Wireless Networking

Different types of mobile devices communicate using different network protocols. Database vendors usually include a number of network protocol supports in their products, for example, GSM, IEEE 802.11, and Bluetooth.

### Security

To overcome the security problems of mobile database applications such as eavesdropping, the mobile database architecture needs to support both username/password authentication and encrypted communication based on the Secure Sockets Layer (SSL) protocol and other popular encryption algorithms.

In spite of the well designed architectural model, there still exist a number of limitations in the integration of the databases with mobile devices.

## Literature Review

Previous research related to database design for mobile applications has involved various mechanisms for avoiding compromise in the performance of the database due to the use of the wireless network. Some of the techniques used involve reducing the number of data exchanged over the wireless network and providing a data cache on the mobile host (Chan, Si, & Leong, 1998). To address the challenges related to maintenance of data consistency for mobile data access, several techniques have been suggested in the literature. These have ranged from transaction management (Mazumdar & Chrysanthi, 1999) and concurrency control for mobile databases (Prabhu, Ray, & Yang, 2004) to replication of mobile databases which are allocated on the fixed network (Budiarto, Harumoto, Tsukamoto, Nishio, & Takine, 1998). It is argued in Budiarto, Nishio, and Tsukamoto (2002) that without replication, mobile databases have a very low availability, and it is shown using simulation that the performance of replication strategies depends on various factors such as network size, mobility, access ratio, and access concentration. Another important issue for mobile databases is how to provide consistent results for location-specific continuous queries, the likes of which may be encountered when navigating road maps using mobile devices. In Gok and Ulusoy (2000), several approaches are compared in terms of relative performance for providing answers to location-dependent queries from mobile users. An analytical model based on the idle replacement policy is described by Hung, Lin, Peng, and Yang (2001) to solve the problem of overflows of visitor location registers for mobile databases. For a detailed discussion on the various issues related to database design for mobile databases with respect to factors such as mobile location data management, transaction processing and broadcast, cache management and replication, query processing, and mobile Web services, the interested reader may refer to Barbara (1999), Madria, Mohania, Bhowmick, and Bhargava (2002), and Yang, Bouguettaya, Medjahed, Long, and He (2003).

## MAIN THRUST

Sixteen Hong Kong brokerage firms are currently improving their productivity and customer satisfaction levels by using the “mBroker” solution offered jointly by Heracle Technologies Limited and Hutchison Telecom, built on the Palm OS platform for Palm handhelds (Lai, Tam, & Lemaitre, 2004). The mBroker solution provides a secure trading platform to remotely access stock

information in real time and to conduct stock trading activities.

## Problem Description

It is time-consuming and labor-intensive for investors to rely on desktop computers or consult brokerage agents to obtain the latest stock quotes or the trading history of any particular stock index, as well as to place an order. Mistakes such as overlooked orders, data entry errors, or delays due to congested telephone lines often occur when dealing with an agent.

It is critical to guarantee high speed and accuracy for this type of situation. Speeding up the order processing and accessing real-time information without compromising accuracy are goals that all brokers strive for to retain their competitive edge.

## The mBroker Solution

mBroker—an innovative, secure, and cost-effective wireless stock trading system with specific design for PDA stock trading—allows PDA users to get stock quotes and trade stocks at any time in any place in the world. More importantly, it is totally secure as it uses the “Hongkong Post Mobile e-Cert” and also Oracle8i. With mBroker, investors can place an order remotely via an intuitive touch screen interface of Palm handhelds, without the need to contact the agent.

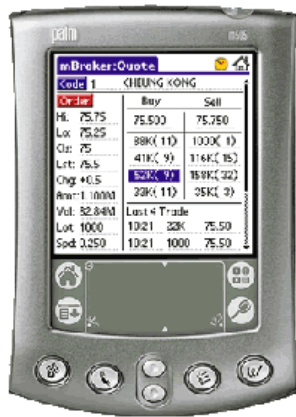
## Functionalities of mBroker

There are a diverse range of services related to stock trading activities provided by mBroker on Palm. These include:

- Place, modify, and cancel orders.
- Access real-time stock quotes.
- Keep track of the Hang Seng Index.
- View order status and transaction history.

The mBroker application can be run on Palm OS 3.5 or above and only occupies approximately 190 Kb of memory space. What makes this solution attractive is the minimal deployment cost. Since there is no special software or hardware requirements for participating brokerage firms, their customers can enjoy the service simply by subscribing to the mBroker service. The user interface is very user-friendly since it has a similar look and feel to any other Palm application. Both English and Chinese versions of this software are available to the users.

Figure 2. Application interface of mBroker on a PDA device



### Architecture of mBroker

Figure 3 shows a simplified view of the mBroker application and the database system architecture. The Oracle database stores stock information and statistics. The stock data includes stock codes, stock names, and stock prices. In order to help the investors in making investment decisions, the database also stores information such as past selling prices of the stock, the high/low price information, and transaction volume statistics. A subset of this information is loaded onto the Palm for fast offline browsing.

Transaction details are also required for order tracking and enquiry. Therefore, all the transaction details are logged in the Palm and the central database. These include the broker ID, status, stock type, price, quantity, transaction timestamp, etc.

The mBroker application follows an architecture that is similar to that discussed earlier. The stock and

transaction information is originally stored in a central Oracle 8i database server. When a mobile device with mBroker wants to retrieve stock quotes from the central database server, it will send requests to a middleware named Oracle Lite Mobile Service in order to subscribe to the information (Viellard, 2001). This middleware is installed in a server machine and acts as a middleman between different mobile devices and the central database server. It obtains the required information from the database server and sends it back to the mobile devices. The middleware is responsible for synchronization, keeping the mobile devices informed of any changes on the database server. When the user buys or sells stock via mBroker, requests are sent from the mobile devices to the middleware. The middleware then updates the central database accordingly. The middleware is capable of receiving thousands of requests at the same time and decides the sequence of processing the transaction requests following predefined business rules written in PL/SQL.

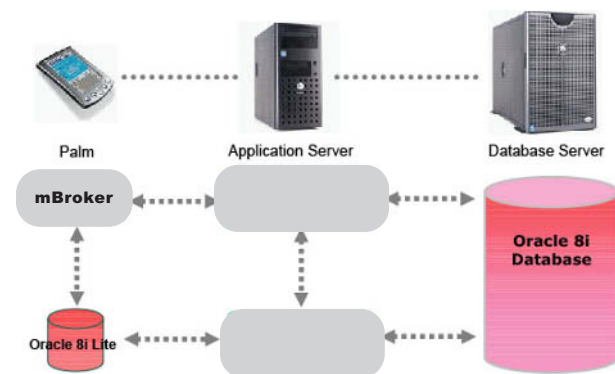
### Choice of Database

Heracle Technologies Limited chose to use Oracle8i as the central component because of mobile service provided by Oracle8i Lite. Oracle8i Lite is accessed by users through the application server and provides the necessary workspace for end users to request context switching between online and offline modes. It also automatically initiates the necessary two-way replication of data and applications between server and client, depending on changes of mode. When it comes to stock order transactions, it is imperative to ensure a maximum level of security. Not all proprietary databases, e.g., DB2, can support a public key infrastructure (PKI; Browder, 2002); Oracle8i supports PKI and therefore this was the obvious choice for mBroker.

### Security of mBroker

To ensure the security of the stock order information (OI), the OI is encrypted using the PIN input and sent to Hongkong Post's Certification Authority server for authentication. All transaction details are secured by Hongkong Post Mobile e-Cert, secured by PKI technology. The Mobile e-Cert can be obtained from Hutchison Telecom, the world's first certified Registration Authority for the issuance of Mobile e-Cert. A high security solution ECC163 cryptography is adopted. Upon verification, the order is allowed to pass through the Order Routing System gateway of Hong Kong Exchanges and Clearing Limited (HKEx) to the designated brokerage firm's system for subsequent processing. Once the transaction is completed, a confirmation note with a

Figure 3. A simplified view of the architecture of mBroker





transaction reference number is sent back to the user's mBroker interface.

### Disaster Recovery in mBroker

Oracle8i database has a component called Data Guard, which offers the most complete and robust disaster recovery solution and high availability through the use of a transactionally consistent standby database. The Data Guard automates the complex tasks of disaster recovery and provides monitoring, alerting, and control mechanisms to maintain a standby operation. Moreover, Data Guard reduces planned downtime by utilizing the standby server for maintenance and routine operations in addition to reporting.

### Current Performance of mBroker

mBroker takes less than six seconds on average to complete an order via a handheld, whereas the user takes approximately one minute to complete the same using a phone call. The issue of wireless security is addressed by the introduction of user/server authentication and digital signatures. As a result, an automated, speedy order placement in a highly secured wireless environment is ensured.

### FUTURE TRENDS

In view of the issues discussed so far it is obvious that there are many areas which need to be improved. In the future, the goal is to provide the following functionalities for mBroker (Huntsman, 2003):

- Providing intelligent roaming capabilities to enable users to work without interruption, even when network connections are disrupted.
- Exploiting multiple network interfaces in a single device or being able to select the fastest or least costly connection.
- Successfully synchronizing databases by caching contents to local devices through asynchronous connections.
- Allowing portability to a range of devices.
- Conserving power at the operating system level and maximizing performance.

### CONCLUSION

In this article we have provided a brief background on the use of databases for mobile applications. This is a grow-

ing area and is facing a number of challenges at this time. The database design for mobile applications is also discussed in this article. We have also discussed a successful mobile application called mBroker which is currently in use in Hong Kong. Another similar example is the new mobile workforce effectiveness solution called SMARTselling, developed by Eleven Technology and powered by SQL Anywhere Studio from iAnywhere Solutions. This technology is currently being used by Pepsi Bottling Group and by Proctor and Gamble. This software helps in automated order entry and runs on a small handheld device that communicates wirelessly with back-office systems. With the help of this application, tedious, error-prone, and costly paper-based processes can be eliminated, and the time spent on checking inventories and shelf displays can be significantly reduced. It is hoped that in the future more mobile applications like mBroker and SMARTselling will be developed, which will affect the various facets of everyday life for people around the globe.

### REFERENCES

- Barbara, D. (1999). Mobile computing and databases: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 11(1), 108-117.
- Browder, K. (2002). *Technical comparison of Oracle database and IBM DB2 UDB: Focus on security*. Retrieved December 2, 2003, from [http://www.oracle.com/ip/se/o9idb\\_db2\\_techcompar.pdf](http://www.oracle.com/ip/se/o9idb_db2_techcompar.pdf)
- Budiarto, K., Harumoto, M., Tsukamoto, M., Nishio, S., & Takine, T. (1998). Replica allocation strategies for mobile databases. *IEICE Transactions on Information and Systems*, E81-D1, (pp. 112-121).
- Budiarto, K., Nishio, S., & Tsukamoto, M. (2002). Data management issues in mobile and peer-to-peer environments. *Data and Knowledge Engineering*, 41, 183-204.
- Chan, B. Y. L., Si, A., & Leong, H. V. (1998). Cache management for mobile databases: Design and evaluation. *Fourteenth International Conference on Data Engineering, ICDE-98*, Orlando, Florida (Vol. 2, No. 7, pp. 54-63).
- Dhawan, C. (1997). *Mobile computing: A systems integrator's handbook*. New York: McGraw-Hill.
- Gok, H. G., & Ulusoy, O. (2000). Transmission of continuous query results in mobile computing systems. *Information Sciences*, 125, 37-63.



Hung, H.-N., Lin, Y.-B., Peng, N.-F., & Yang, S.-R. (2001). Resolving mobile database overflow with most idle replacement. *IEEE Journal on Selected Areas in Communication*, 19(10), 1953-1961.

Huntsman, J. B. (2003). *Introducing the Intel mobile application architecture guide*. Retrieved December 2, 2003, from <http://www.intel.com/update/contents/sw12031.htm>

Lai, A., Tam, A., & Lemaitre, S. (2004). *Hong Kong stock trading industry embarks on a new era with mBroker capabilities on Palm OS platform*. Retrieved December 2, 2003, from <http://www.dvnet.com/pdf/casestudies/palm.pdf>

Madria, S. K., Mohania, M., Bhowmick, S. S., & Bhargava, B. (2002). Mobile data and transaction management. *Information Sciences*, 141, 279-309.

Mazumdar, S., & Chrysanthis, P. K. (1999). Achieving consistency in mobile databases through localization in PRO-MOTION. *Second International Workshop on Mobility in Databases and Distributed Systems (MDDS99)*, Florence, Italy (pp. 82-89).

Munusamy, M., & Hiew, P. L. (2004). *Characteristics of mobile devices and an integrated m-commerce infrastructure for m-commerce deployment*. Retrieved December 2, 2003, from <http://www.wayneyeung.com/files/papers/FP-102.pdf>

Prabhu, N., Kumar, V., Ray, I., & Yang, G.-C. (2004, March). Concurrency control in mobile database systems. *18<sup>th</sup> International Conference on Advanced Information Networking and Application (AINA04)*, 2, 83-86.

Viellard, E. (2001). *Oracle9i Lite Business White Paper*. Retrieved December 2, 2003, from [http://otn.oracle.com/products/lite/pdf/o9ilite\\_bwp.pdf](http://otn.oracle.com/products/lite/pdf/o9ilite_bwp.pdf)

Yang, X., Bouguettaya, A., Medjahed, B., Long, H., & He, W. (2003). Organizing and accessing Web services on air. *IEEE Transactions on Systems, Man, Cybernetics, Part A: Systems and Humans*, 33(6), 742-757.

## KEY TERMS

**Bluetooth:** A wireless technology developed by Ericsson, Intel, Nokia, and Toshiba that specifies how mobile phones, computers, and PDAs interconnect with each other, with computers, and with office or home phones. The technology enables data connections between electronic devices in the 2.4 GHz range. Bluetooth

can replace cable or infrared connections for such devices.

**Caching:** The technique of copying data from a server machine (the central storage place) to a client machine's local disk or memory; users then access the copy locally. Caching reduces network load because the data does not have to be fetched across the network more than once (unless the central copy changes).

**Database Synchronization:** When a database is being synchronized, no new update transactions are allowed, and all open update transactions are finished. After that, all updated blocks are written to disk.

**Horizontal Industry Applications:** A horizontal industry is one that aims to produce a wide range of goods and services. Horizontal industry applications are utilized across many different industries. While the core part of the application does not require changes, an organization needs customization at the front end or at the back end. Database access and service representative dispatch are typical examples of these applications.

**IEEE 802.11:** 802.11 refers to a family of specifications developed by the IEEE for wireless LAN technology. 802.11 specifies an over-the-air interface between a wireless client and a base station or between two wireless clients. The IEEE accepted the specification in 1997.

**Load Balancing:** It is the method of distributing system load evenly across server machines by placing identical copies of frequently accessed information among available server machines.

**Middleware:** This software manages the communication between a client program and a database. For example, a Web server connected to a database can be considered middleware as the Web server sits between the client program (a Web browser) and a database. The middleware allows the database to be changed without necessarily affecting the client and vice versa.

**Mobile Application:** A mobile application is any application that can be used on the move. It may or may not be wireless. It must be tailored to the characteristics of the device that it runs on. Limited resources, low network bandwidth, and intermittent connectivity are all important factors that affect the design of these applications.

**Mobile Device:** A mobile device is anything that can be used on the move, ranging from laptops to mobile phones. As long as the location is not fixed, it is considered mobile. Areas that are not included in the definition of mobile include remote offices, home offices, or home appliances.

**Public Key Infrastructure (PKI):** A system that enables users of a public network to exchange data securely and privately through the use of a public and private cryptographic key pair, which is obtained and shared through a trusted authority. It provides for a digital certificate that can identify an individual or an organization and director services that can store and, when necessary, revoke the certificates. The comprehensive architecture includes key management, the registration authority, certificate authority, and various administrative tool sets.

**Replication:** The process of creating read-only copies of any data. Replication is supported by the security, directory, and file services in a distributed computing environment. Replication can improve availability and load balancing.

**Secure Sockets Layer (SSL):** SSL is a transaction security standard developed by Netscape Communications to enable commercial transactions to take place over the Internet. It's one of a few competing security standards.

**Vertical Industry Applications:** A vertical industry is one that is focused on a relatively narrow range of goods and services. Vertical industry applications are specific to certain industries. Usually there are some characteristics of the business processes unique to a particular industry that make certain applications very specific for that particular industry. As a result, some vendors develop turn-key software solutions for their own use.

# Dataveillance and Panoptic Marketspaces

**Nikhilesh Dholakia**

*University of Rhode Island, USA*

**Detlev Zwick**

*York University, Canada*

**Anil Pandya**

*Northeastern Illinois University, USA*

## ELECTRONIC MARKETS

With continuing global growth of electronic commerce, consumers have come to exist in electronic “marketspaces” (Rayport & Sviokla, 1994) that are frequently surveilled through database techniques. This type of data-driven monitoring has been characterized as “dataveillance.”

This article reviews the main characteristics of such dataveilled and panoptic marketspaces, from the perspectives of critical social theories.

## EXAMPLES OF DATAVEILLED MARKETSPACES

The term *dataveillance* is attributed to Australian computer scientist Roger Clarke (1988). Apart from the technically informed critiques by Clarke, the virtual dataspaces intersecting with human bodies have been discussed by Kroker and Weinstein (1994) and the condition of humans located in database matrices has been critiqued by Poster (1990a, 1995a). Aspects of dataveillance in e-commerce and m-commerce marketspaces have been addressed by Dholakia and Zwick (2001) and Zwick and Dholakia (2004).

What do the emergent dataveilled marketspaces look and feel like? We provide short descriptive cases to illustrate them.

### JetBlue Passengers Face the Blues

The U.S. Department of Defense carried out a project to identify potential terrorists among airline passengers who could possibly attack army bases. Ordinary airline passengers became the unwitting pawns in this data-invasion exercise. In September, 2002, Torch Concepts—a data mining firm on contract from the army—acquired from JetBlue Airways (a discount air carrier) the itinerary

data for over 1.5 million passengers, including passenger names, addresses, and phone numbers. Of these passenger records, 40% were cross-referenced with gender, home specifics (owner/renter, etc.), years at residence, economic status (income, etc.), number of children, Social Security number, number of adults, occupation, and vehicle information—detailed demographic data obtained from a database vendor. While this substantial panoptic exercise—undertaken without any knowledge or consent of the passengers concerned and in violation of JetBlue Airways declared privacy policies—found only one possible anomalous passenger record, the potential for massive invasion of marketspace data records is quite evident in this case (EPIC, 2003).

### Do the Insured Have a Clue?

Michael Ha of National Underwriter is worried that when massive databases such as the Department of Motor Vehicles (DMV) records and Comprehensive Loss Underwriting Exchange (CLUE) cross-reference each other, insurance companies can benefit at the expense of clueless drivers. Although such databases have legitimate uses, such as determining insurance premium rates for motorists, they can also be misused. Blending such private data with data collected from emerging technologies such as event recorders and global positioning systems, these systems are capable of generating real-time data profiles of drivers, in effect providing insurers a secretive “eye in the sky” to monitor drivers. When such data is married with information in massive DMV and CLUE databases, serious privacy concerns arise. Some insurance companies are encouraging automobile makers to install monitoring technologies in high-end vehicles. Ethical boundaries are being crossed, however. Allstate, a major insurer, has already paid \$1 million to the California DMV in settlement of a dispute about unethical use of DMV records (Ha, 2003).

## **Your CEO Wants You to Do This**

If you are assisting your CEO with forecasting tasks, how would you react to a personalized letter, such as this from a software maker:

Mr. P [actual name of your CEO] is interested in this product. What would it be worth to Mr. P if *you* increase the accuracy of your decisions by 10%? How about 25%? What if you double your effectiveness? Although the future is difficult to foresee, I predict your next “What-If” analysis will result in better insights. Your superiors such as Mr. P will place more confidence in your analysis. You’ll make better decisions. (Brent Green & Associates, 2003)

Such a letter was sent by database marketing firm Brent Green & Associates (BGA) on behalf of the software firm making the forecasting product. The letter was “signed” by the president of this software firm, and BGA selected a list of subscribers to a well-known newsletter targeting financial planners, security investment advisors, security analysts, and brokers. Then BGA matched each prospect on the first list with another list containing names of company presidents obtained from American Business Information, which also included company details. Based on such matched names, each prospect received a letter from BGA bearing the message portrayed above. Some of the letter recipients were enraged at this invasion of privacy, and the president of BGA received a few flaming telephone calls. One caller was livid and threatened to notify the Better Business Bureau and promised that his bank would never buy software from this company. Thus, BGA learned that marrying two or more databases is risky, even if creative overlays offer tempting ways to grab attention (Brent Green & Associates, 2003).

## **Have You Refilled Your Prescription?**

Rite Aid Pharmacy wrote a letter, sponsored by a pharmaceutical manufacturer, to Lisa in California to “remind” her to refill a particular prescription medication that she was taking. Shortly thereafter, Rite Aid called her husband to “remind” him to refill a particular prescription medication that he was taking. The woman felt that Rite Aid was using her prescription records and her family’s private medical information to aggressively market prescription medications. She considered the conduct of Rite Aid unprofessional and complained to the California Department of Consumer Affairs (Privacy Rights Clearinghouse, 1999).

## **PRIVACY-PERMISSION DYNAMICS**

With increased dataveillance comes increasingly data-intrusive marketing techniques, such as spam e-mail. Publicly accessible databases and data streams might be easily harvested by aggressive spammers, often for unscrupulous use. In the deluge of unwanted spam and pop-up windows, legitimate and desired marketing interactions could get sidelined. Permission marketing (Godin, 1999), with its very tightly specified “opt-in” e-communication parameters, and e-commerce Web sites, with strong privacy policies, are belated responses to such developments.

In principle, permissions-based e-commerce seems to take care of privacy invasion problems. In practice, things are more complex. In the evolving privacy-permission dynamics in electronic marketspaces, several contentious issues remain. For example,

- Opt-out permission methods, capable of confusing all but the most vigilant consumers, are used far more frequently than consumer-friendly opt-in methods.
- The scope of permission—its breadth and depth—is often unclear, and companies can take advantage of this by interpreting permissions in broader and deeper ways that favor them rather than the consumers.
- It is unclear whether permission granted by the customer to a marketer is transferable to a third party that acquires the customer’s information from the marketer.
- Customer permission is very business-specific. What happens if the business is acquired and the customer does not want to grant permission to the acquiring company?
- Conversely, opt-out decisions may not be honored by the new owner of the original customer database, thus invalidating the customer’s initial decision.

## **CRITICAL SOCIAL THEORY PERSPECTIVES**

Inspired by philosopher of technology Mark Poster (1990b; 1995b), we develop a poststructuralist critique of customer database technology. This approach construes information technology such as databases as configurations of language that produce new and signifi-

cant social, cultural, and political representations. The mode of representation of databases is unique because it transforms the items that databases absorb into specific codes. Increasingly, databases absorb consumers, affecting how they are configured by organizations as “customers” and how organizations relate to these digital customer representations.

We analyze the ideological and cultural work that information technology and databases do in organizations and markets. We do not, however, concern ourselves with the technical and computer-scientific aspects of databases and other information systems.

At the center of our discussion are computerized customer databases. Approaching information technology from a poststructuralist perspective, we call attention to the *discursive effects* of databases, in Foucault’s sense. Foucault’s understanding of discourse and language is of special relevance for our discussion because of the relation he draws between language and the constitution of the subject. On this view, a new language introduces a new way of constituting cultural objects (e.g., human beings, the organization, customers, suppliers) and social reality (e.g., markets, relationships). Computerized databases constitute such a new language, and their introduction has changed the organizational language for understanding the market in general and customers in particular.

Along with the rise of database technologies and electronically mediated languages, new forms of institutional and organizational power have emerged. Two camps, one with a libertarian and the other with a Marxist outlook, have been commenting forcefully on the implications of this new “mode of information” (Poster, 1990b). The libertarians are mostly concerned about the potential centralization of surveillance power in the hands of state and commercial institutions. On this view, database technology and networked communications are regarded as additional components of the always-growing threat of overbearing government and evermore intrusive corporations. For the libertarians, the biggest fear is that databases brimming with detailed personal information will hasten the end of privacy for citizens and of free choice for consumers (Dholakia & Zwick, 2001).

Marxists are more concerned about the contribution of computer networks and database technology to the dominance of corporations over the working class. Because information is not equally available to all, this camp maintains that capitalists can seize information technologies to further monopolize control over the means of production. In the information economy, such control includes knowledge workers as well as manual workers. Such a position suggests that management mobilizes the rhetoric of efficiency to obtain workers’ acceptance for erecting a fully transparent organization, where

all production processes are always under surveillance, every member is monitored, and his or her contribution is assessed constantly.

Both viewpoints have much to offer for our understanding of the impact of information technology on personal freedom, consumer sovereignty, and the power of the worker. Yet, they fail to grasp the *cultural innovations* brought about by the integration of information technologies, such as databases, into existing political, economic, and social institutions (Poster, 1990b, 1995b; Sotto, 1997). The problem of information technology’s cultural effects escapes Marxists and liberal thinkers because they theorize the social field primarily as one of action, minimizing the importance of language. Yet, databases are made up of symbols in data fields. They embody a specific mode of representing the world—what Bolter (2001) calls “numeric inscription.” As Poster (1995) puts it, “One does not eat them, handle them, or kick them, at least one hopes not. Databases are configurations of language; the theoretical stance that engages them must take at least this ontological fact into account.”

The use of information technology to transform citizens, workers, and consumers into digital representations is relevant because it enables new forms of surveillance, control, and management of these populations. Before the arrival of the electronic database and the widespread use of electronic and digital transaction formats in the marketplace, the consumer was not subjected to a permanent and ubiquitous regime of surveillance and observation. The “massified” consumer was able to slip in and out of markets, easily and anonymously. The database, however, is like an air traffic control tower that presents a screen of almost total visibility. On this visual screen, the individual customer is distributed relative to other customers, according to a system of similarities and differences operating in two steps: first, the encoding, and second, the interpretation of details of personal attributes and conduct. The customer is no longer “lost in the fleeting passage of time, space, movement, and voice but identifiable and notable” (Rose, 1988). Within the regime of norms and requirements that make up this perceptual surface, each customer—encoded according to a clearly defined language of data fields and algorithms—can now be placed *in relation to* other customers. The electronic and digital transaction space between organizations and their customers becomes a map, which enables the emergence of what was previously imperceptible and below the threshold of description: the individual customer.

Thus, with the increases in customer database profile sizes, the threshold for detecting ordinary individuality has been continuously lowered. Therefore, it might be useful to think about the mode of functioning of the



customer database as a representational or discursive machine that identifies and harnesses human difference. In this respect, customer databases are conceptually closely related to Foucault's notion of the Panopticon. Foucault developed this idea from Jeremy Bentham's late 18th-century description of an architectural structure that depicted a circular-type prison of individual cells, arranged around the perimeter. A tower rises in the center of the prison, allowing a single warden to see any cell at any time. Foucault regarded the Panopticon as a model for the transition from a traditional to a modern form of society. For him, "this enclosed, segmented space, observed at every point, in which individuals are inserted in a fixed place, in which the slightest movements are supervised, in which all events are recorded, in which an uninterrupted work of writing links the center and periphery...in which each individual is constantly located, examined, and distributed among the living beings" becomes the place where the new technology of observation and recording creates "new truths" about the individual. The Panopticon is therefore less a force of repression and oppression than an inscription and discursive fabrication of individual.

The Panopticon, as a metaphor for our surveillance society, denotes an accumulation and centralized knowledge in the hands of the technology holder. In these databases, consumers take discernible shape for data mining purposes, where they are configured as data-linguistic constructs that can be identified and acted upon strategically. Today, the recording of all events and the databases they generate constitute a Superpanopticon, a system of surveillance without walls, windows, towers, or guards (Poster, 1990). Technological advances, however, are only part of the story. The consumer has become accustomed to surveillance and even to participating in the process of data accumulation. Social Security numbers, credit cards, library cards, drivers' licenses, and loyalty cards—the consumer must apply for them, use them continuously, have them ready at all times, and verify their validity by cross-referencing them with other documents. Each transaction is recorded, encoded, and added to the database. The circle of seamless and continuous dataveillance has been closed.

Thus, though consumers *participate* in the formation and population of their own data records by committing simple consumption acts, they do not *completely control* this (in)formation. In fact, with online tracking, powerful recoding and filtering technologies, and more frequent selling and exchanging of customer records, consumers are no longer able to always decide what kind of information about them is stored, categorized, manipulated, exchanged, and acted upon by whom, when, and where. What is more important, matters of

control over one's personal information are complicated by the emergence of new information technologies. Radio Frequency Identification (RFID) technologies, for example, can be used for smart tags. RFID tags are used to track assets, manage inventory, and authorize payments, and they increasingly serve as electronic keys for everything from automobiles to secure facilities. The European clothier Benetton was among the first consumer product manufacturers to equip merchandise such as pants and sweaters with RFID tags by weaving them into the traditional garment tag. Though the tag does not directly store information about the person wearing the Benetton product, it nevertheless has the potential to communicate much about him or her. Consider a person wearing Benetton pants equipped with an RFID tag during a shopping trip in a downtown mall. Inconspicuous receivers placed in stores and public spaces could continuously communicate with the pant's smart tag, eventually painting a detailed picture about the person's shopping trajectory.

In sum, databases form the basis of the panoptic marketspace. The act of consumer individualization—based on the workings of the database—transforms the ways in which inscriptions of human individuality can be produced, ordered, accumulated, and circulated. In its essence, database-driven information technologies "provide a technique of visualization and inscription of individuality which objectify their subjects by inscribing their differences from one another" (Rose, 1988, p. 195). The ways in which consumers are dataveilled, fabricated as data objects, and acted upon to gain control over their behaviors need to be understood in order to develop strategies to counteract the imbalance of panoptic power. The accumulation and centralization of knowledge about the consumer in customer databases make the database the site where customers are constituted as culturally relevant objects of analysis, knowledge, and managerial action. Any form of political action that aims to curtail the power of the panoptic marketspace must take place at the level of the database.

## **POLICY IMPLICATIONS**

With customer databases growing and becoming ubiquitous, consumers are becoming increasingly known and acted upon as digital representations. Hence, the constitution of the customer as epistemological entity takes place within the code of the database. In addition, it is increasingly impossible for consumers to prevent dataveillance by marketers or to participate in the formation of their own data profiles. Therefore, actions that go beyond the generation and application of individual information externalization strategies are needed. Indeed, we argue that consumers must be given direct access to customer databases to ensure that they regain a viable

voice in the process of their constitution as digitized customers.

Always at the leading edge of database marketing, Amazon.com has come to understand that constituting valuable and strategically superior digital consumer identities cannot be accomplished without customers' direct access to its databases. After recognizing return customers via sign-ins and cookies, Amazon.com offers them dozens of product recommendations. After each recommendation, the hyperlink "Why was I recommended this?" is offered. Clicking on this hyperlink offers a glimpse into the section of the Amazon database that stores previous, correlated purchases of the customer. The customer then has options of clicking on "Not Interested" in that specific recommendation or to intensify or dilute the recommendation algorithm by tweaking the intensity of preference for the previously purchased, correlated items on a 5-point scale. In effect, Amazon.com lets consumers take partial control of the constitution of their customer identity by allowing them to *access and alter the code and content of the database*.

Similarly, in Canada the privacy commissioner of the Province of Ontario has announced an initiative that will open up all government databases to its citizens, enabling them to make changes to the data profile currently stored there. Lyotard (1984) proclaimed some time ago that we must "give the public free access to the memory and data banks" (p. 67). Twenty years later, his call may finally be heeded.

## CONCLUSION

Dataveilled consumers constitute digitized customer representations in vast databases—this is the reality of contemporary business in advanced economies. Incentives and sanctions are in place to ensure that consumers participate—willingly or reluctantly—in this ongoing accumulation of data.

For business, consumers, and public policymakers, there are serious challenges ahead. In panoptic, digitized customer databases, there are many possibilities of privacy invasion, misrepresentation, identity theft, and other social ills. Therefore, from private business-strategic perspectives as well as enlightened public policy perspectives, it is necessary take actions that go beyond the generation and application of individual information externalization strategies. Customer databases will not go away—they will only increase in size and sophistication with the advent of new technologies such as RFID tags and biometrics. Therefore, enlightened policies should ensure that consumers have direct access to customer databases and that they have some say in the processes of their constitution as digitized customers.

## REFERENCES

- Bolter, J. D. (2001). *Writing space: Computers, hypertext, and the remediation of print*. Mahwah, NJ: Erlbaum.
- Brent Green Associates. (2003). <http://bgassocia tes.com/essay2.htm>
- Clarke, R. (1988). Information technology and dataveillance. *Communication of ACM*, 31(5), 498-512.
- Dholakia, N., & Zwick, D. (2001). Privacy and consumer agency in the information age: Between prying profilers and preening Webcams. *Journal of Research for Consumers*, 1(1), Article 3. Retrieved from <http://www.jrconsumers.com>
- EPIC. (2003). [http://www.epic.org/privacy/airtravel/nwa\\_comp.pdf](http://www.epic.org/privacy/airtravel/nwa_comp.pdf)
- Godin, S. (1999). *Permission marketing: Turning strangers into friends, and friends into customers*. New York: Simon & Schuster.
- Ha, M. (2003, April 21). Consumer groups try to push big brother insurer out of passenger seat. *National Underwriter (Property & Risk Management Edition)*, 107(16).
- Kroker, A., & Weinstein, M. A. (1994). *Data trash: The theory of the virtual class*. Montreal, Canada: New World Perspectives.
- Lyotard, J.-F. (1984). *The postmodern condition: A report on knowledge: Vol. 10* (G. Bennington & B. Massumi, Trans.). Minneapolis: University of Minnesota Press.
- Poster, M. (1990a). Foucault and databases. *Discourse*, 12(2), 110-127.
- Poster, M. (1990b). *The mode of information*. Chicago: The University of Chicago Press.
- Poster, M. (1995a). Databases as discourse, or electronic interpellations. In P. Heelas, S. Lash, & P. Morris (Eds.), *Detraditionalization* (pp. 277-293). Oxford, UK: Blackwell.
- Poster, M. (1995b). *The second media age*. Cambridge, MA: Polity Press.
- Privacy Rights Clearinghouse. (1999). <http://www.privacyrights.org/>
- Rayport, J. F., & Sviokla, J. J. (1994). Managing in the marketplace. *Harvard Business Review*, 72(6), 141-150.

## ***Dataveillance and Panoptic Marketspaces***

Rose, N. (1988). Calculable minds and manageable individuals. *History of the Human Sciences*, 1(2), 179-200.

Sotto, R. (1997). The virtual organization. *Accounting, Management and Information Technology*, 7(1), 37-51.

Zwick, D., & Dholakia, N. (2004). Whose identity is it anyway? Consumer representation in the age of database marketing. *Journal of Macromarketing*.

### **KEY TERMS**

**Dataveillance:** The monitoring of people by digital representations in electronic databases created and managed by information technologies.

**Marketspace:** Electronic transaction methods, or electronic markets, in which businesses and consumers interact.

**M-Commerce:** Mobile commerce, transactions on mobile phones. Examples include, in addition to voice communications, SMS messages, game downloads, stock market quotes, and the like.

**Opt-In & Opt-Out:** Options for individuals to be included or excluded from a database record

**Panopticon:** Jeremy Bentham's late 18th century description of an architectural structure that depicted a circular-type prison of individual cells arranged around the perimeter. A tower rises in the center of the prison allowing a single warden to see any cell at any time. Foucault popularized the use of Panopticon and panoptic constructs as ways of characterizing dataveilled social spaces.

**Pop-Ups:** Messages and advertisements that show up on one's computer screen without permission

**RFID:** Radio frequency identification that uses low-powered radio transmitters to read data stored in a transponder (tag) at distances ranging from 1 in. to 100 ft. RFID tags are used to track assets, manage inventory, and authorize payments, and they increasingly serve as electronic keys for everything from automobiles to secure facilities.

**D**

# Deriving Spatial Integrity Constraints from Geographic Application Schemas

**Clodoveu A. Davis, Jr.**

*Pontifícia Universidade Católica de Minas Gerais and Empresa de Informática e Informação do Município de Belo Horizonte, Brazil*

**Karla A. V. Borges**

*Empresa de Informática e Informação do Município de Belo Horizonte and Universidade Federal de Minas Gerais, Brazil*

**Alberto H. F. Laender**

*Universidade Federal de Minas Gerais, Brazil*

## INTRODUCTION

Integrity constraints of various kinds must be observed when creating or updating a database in order to preserve the semantics and the quality of stored data (Elmasri & Navathe, 2000). Within the scope of geographic applications, integrity assurance requires special attention from the designer, since most geographic applications use data that depend on spatial relationships (Egenhofer & Franzosa, 1991), thereby requiring the specification of *spatial integrity constraints*.

In the traditional database approach, there is a relationship between conceptual, logical, and physical design, in which, through mapping operations, constraints that are identified in the conceptual schema are inherited and transformed into implicit constraints expressed by the data definition language (DDL) or into explicit constraints coded in the application programs (Elmasri & Navathe, 2000). This relationship also exists in spatial information systems; therefore, spatial constraints can be likewise identified and implemented. However, even though there is a very active research area interested in the design of robust and efficient spatial databases, there are still shortcomings with respect to spatial integrity constraints (Borges, Davis & Laender, 2002; Plumber & Groger, 1997). This happens mostly because a simple geometric modification in a single object in a spatial database may generate the need to check for possible integrity violations throughout many object classes, using computationally intensive geometric and topologic algorithms.

Most spatial integrity constraints are, in fact, semantic integrity constraints applied to the spatial representation of objects and to the relationships among object instances that are based on spatial representations. In order to be able to adequately represent such representa-

tions and relationships in geographic applications design, tools that are more specific and capable of capturing the semantics of geographic data, offering higher abstraction mechanisms and implementation independence (Borges, Davis & Laender, 2001; Câmara, 1995) are required. From geographic application database schemas, developed using an adequate data model, representations and spatial relationships can be extracted; thus, spatial integrity constraints can be specified.

This paper focuses on the types of spatial integrity constraints that derive from spatial data modeling constructs, as a part of spatial databases design. OMT-G (Borges et al., 2001), an object-oriented data model for geographic applications, is used to illustrate the concepts involved in the definition of such constraints.

## BACKGROUND

Every data model has a set of built-in constraints associated with its constructs (Elmasri & Navathe, 2000). Accordingly, the OMT-G model allows several spatial integrity rules to be derived from its primitives. These rules constitute a set of constraints that must be observed in the operations that update a geographic database. Ideally, these rules should be implemented by the spatially-enabled database management system (DBMS) in its data definition language (DDL), along with the necessary expansions to the data manipulation language (DML) and the implementation of geographic data types and access methods. However, the approach employed by current commercial products is rather different. Since current spatial DBMSs do not implement spatial integrity constraints, the task of ensuring consistency ends up in the application, usually developed as a geographic information system (GIS). In general, GIS tools allow inconsistent

information to enter the database; later on, a range of correction functions is used to “clean up” the data, verifying its consistency.

Regardless of whether the implementation of integrity constraints is to be performed by the DBMS or by the GIS, the required constraints can be determined at the conceptual level of spatial databases design. In order to show how this can be done, we will use conceptual schema primitives from the OMT-G model (Borges et al., 2001), showing how the semantics embedded in the primitives leads to the spatial integrity constraints. Such an exercise can be performed on any other spatial data model.

### SPATIAL INTEGRITY CONSTRAINTS AND THE OMT-G MODEL

OMT-G is a conceptual data model based on the primitives of the UML class diagram (Rational, 1997), enhancing it with geographic primitives. It is based on three main concepts: *classes*, *relationships*, and *spatial integrity constraints*. Classes and relationships are the basic primitives that are used to create application schemas with OMT-G. From these primitives, spatial integrity constraints can be obtained, as presented next.

*Classes*, in OMT-G, represent the three main groups of data (continuous, discrete, and non-spatial) that can be found in geographic applications. Non-spatial data are represented using *conventional classes*, which can relate to spatial objects but have no geometric or geographic properties. *Georeferenced classes* describe a set of objects that have spatial representation and are associated to features on Earth (Câmara, 1995). Assuming the fields and objects view (Frank & Goodchild, 1990), georeferenced classes are specialized into *geo-field* and *geo-object* classes. Geo-field classes represent objects and phenomena that are continuously distributed over the space, corresponding to variables such as soil type,

relief, and mineral contents. Geo-object classes represent individual, particular geographic objects, which can usually be traced back to real-world elements, such as buildings, rivers, and trees. The notation employed by OMT-G is shown in Figure 1.

There are five geo-field descendant classes in OMT-G: isoline, planar subdivision, tessellation, sampling, and triangular irregular network (Figure 2). From the semantics involved in the concept of geo-fields and from the specific definition of these classes, some spatial integrity rules can be deduced (Table 1).

Geo-object classes are classified into *geo-object with geometry* classes, representing objects which have only geometric properties (points, lines, and polygons), and *geo-object with geometry and topology*, which represents objects which also have topological connectivity properties, and are thus specifically suited to the representation of spatial network structures. Topological properties are present in objects that are either nodes or arcs in a graph-theoretic approach, thereby forming *Node*, *Unidirectional Line*, and *Bidirectional Line* classes (Figure 3). Geo-objects with geometry and topology are not subject to a set of integrity constraints by themselves, but their use is conditioned to the existence of network relationships (Table 4).

The geometric concepts used in the definition of points, lines (including lines with a topological role), and polygons lead to some integrity constraints. The geometric definitions adopted in OMT-G admit the existence of geo-objects that are formed by several polygons, establishing one of them as the “basic” polygon and considering the others as *islands* or *holes*. Polygons that are composed of multiple parts (or *polygonal regions*) are important, since there is no guarantee that the results of traditional operations, such as buffer creation, union, intersection, and difference between simple polygons, are always formed with simple polygons. Constraints regarding lines and polygons are presented in Table 2.

Figure 1. Graphic notation for the basic classes

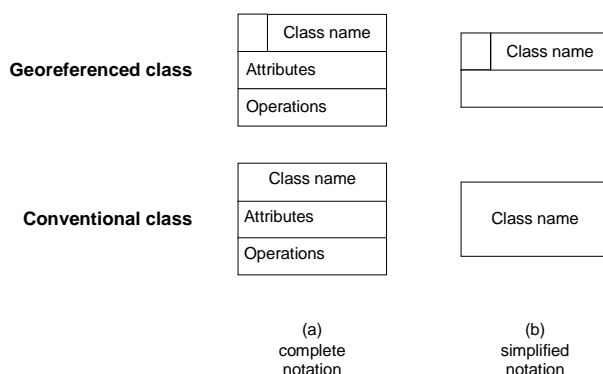


Figure 2. Geo-field classes

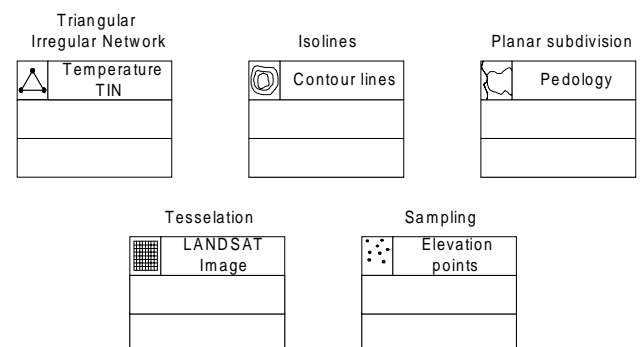




Table 1. Geo-field integrity rules

<b>Planar Enforcement Rule</b>	1. Let $F$ be a geo-field and let $P$ be a point such that $P \subset F$ . Then a value $V(P) = f(P, F)$ , i.e., the value of $F$ at $P$ , can be univocally determined.
<b>Isoline</b>	2. Let $F$ be a geo-field. Let $v_0, v_1, \dots, v_n$ be $n+1$ points in the plane. Let $a_0 = \overline{v_0v_1}, a_1 = \overline{v_1v_2}, \dots, a_{n-1} = \overline{v_{n-1}v_n}$ be $n$ segments, connecting the points. These segments form an <i>isoline</i> $L$ if, and only if, (1) the intersection of adjacent segments in $L$ is only the extreme point shared by the segments (i. e., $a_i \cap a_{i+1} = v_{i+1}$ ), (2) non-adjacent segments do not intercept (that is, $a_i \cap a_j = \emptyset$ for all $i, j$ such that $j \neq i+1$ ), and (3) the value of $F$ at every point $P$ such that $P \in a_i, 0 \leq i \leq n-1$ , is constant.
<b>Tessellation</b>	3. Let $F$ be a geo-field. Let $C = \{c_0, c_1, c_2, \dots, c_n\}$ be a set of regularly-shaped cells covering $F$ . $C$ is a <i>tessellation</i> of $F$ if and only if for any point $P \subset F$ , there is exactly one corresponding cell $c_i \in C$ and, for each cell $c_i$ , the value of $F$ is given.
<b>Planar Subdivision</b>	4. Let $F$ be a geo-field. Let $A = \{A_0, A_1, A_2, \dots, A_n\}$ be a set of polygons such that $A_i \subset F$ for all $i$ such that $0 \leq i \leq n-1$ . $A$ forms a <i>planar subdivision</i> representing $F$ if and only if for any point $P \subset F$ , there is exactly one corresponding polygon $A_i \in A$ , for which a value of $F$ is given (that is, the polygons are non-overlapping and cover $F$ entirely).
<b>Triangular Irregular Network</b>	5. Let $F$ be a geo-field. Let $T = \{T_0, T_1, T_2, \dots, T_n\}$ be a set of triangles such that $T_i \subset F$ for all $i$ such that $0 \leq i \leq n-1$ . $T$ forms an <i>triangular irregular network</i> representing $F$ if and only if for any point $P \subset F$ , there is exactly one corresponding triangle $T_i \in T$ , and the value of $F$ is known at all of vertices of $T_i$ .

Figure 3. Geo-object classes

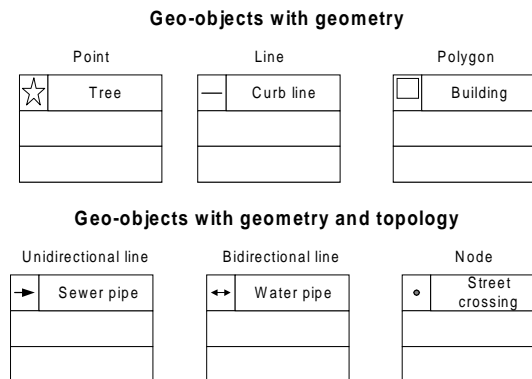
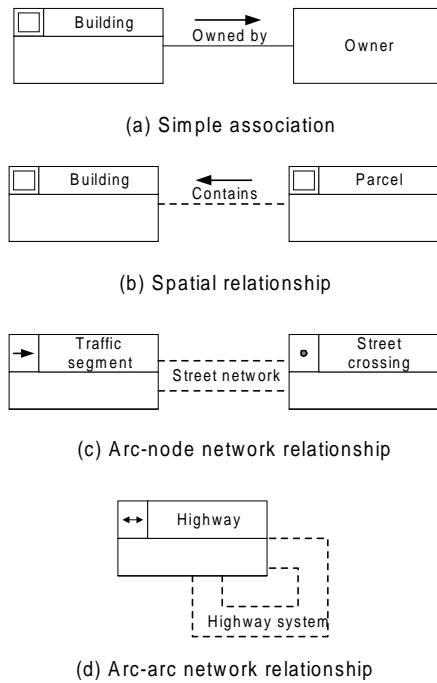


Table 2. Geo-object constraints

<b>Line</b>	1. Let $v_0, v_1, \dots, v_n$ be $n+1$ points in the plane. Let $a_0 = \overline{v_0v_1}, a_1 = \overline{v_1v_2}, \dots, a_{n-1} = \overline{v_{n-1}v_n}$ be $n$ segments, connecting the points. These segments form a <i>polygonal line</i> $L$ if, and only if, (1) the intersection of adjacent segments in $L$ is only the extreme point shared by the segments (i. e., $a_i \cap a_{i+1} = v_{i+1}$ ), (2) non-adjacent segments do not intercept (that is, $a_i \cap a_j = \emptyset$ for all $i, j$ such that $j \neq i+1$ ), and (3) $v_0 \neq v_{n-1}$ , that is, the polygonal line is not closed.
<b>Simple Polygon</b>	2. Let $v_0, v_1, \dots, v_{n-1}$ be $n$ points in the plane, with $n > 3$ . Let $s_0 = \overline{v_0v_1}, s_1 = \overline{v_1v_2}, \dots, s_{n-2} = \overline{v_{n-2}v_{n-1}}$ be a sequence of $n-1$ segments, connecting the points. These segments form a <i>simple polygon</i> $P$ if, and only if, (1) the intersection of adjacent segments in $P$ is only the extreme point shared by the segments (i.e., $s_i \cap s_{i+1} = v_{i+1}$ ), (2) non-adjacent segments do not intercept (i.e., $s_i \cap s_j = \emptyset$ for all $i, j$ such that $j \neq i+1$ ), and (3) $v_0 = v_{n-1}$ , that is, the polygon is closed.
<b>Polygonal Region</b>	3. Let $R = \{P_0, P_1, \dots, P_{n-1}\}$ be a set formed by $n$ simple polygons in the plane, with $n > 1$ . Considering $P_0$ to be a basic polygon, $R$ forms a <i>polygonal region</i> if, and only if, (1) $P_i \cap P_j = \emptyset$ , for all $i \neq j$ , (2) polygon $P_0$ has its vertices coded in a counterclockwise fashion, (3) $P_i$ disjoint $P_j$ (see Table 3) for all $P_i \neq P_0$ in which the vertices are coded counterclockwise, and (4) $P_0$ contains $P_i$ (see Table 3) for all $P_i \neq P_0$ in which the vertices are coded clockwise.

Figure 4. Relationships



OMT-G includes three basic *relationship types* (*simple associations, spatial relations, and network relations*) (Figure 4), along with object-oriented relationships (*generalization/specialization, aggregation and conceptual generalization*).

*Simple associations* represent structural relationships between objects of different classes, conventional as well as georeferenced. Integrity constraints corresponding to simple associations are, thus, essentially the same constraints usually implemented in conventional databases.

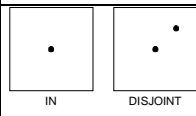
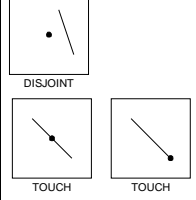
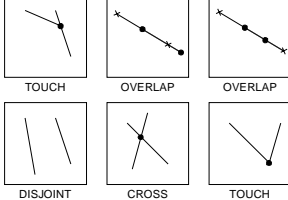
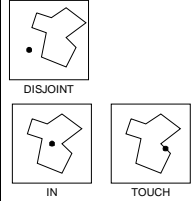
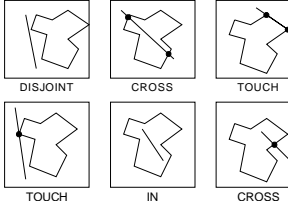
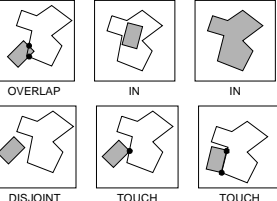
*Spatial relations* represent topologic, metric, ordinal, and fuzzy relationships. Some relations can be derived automatically, from the geometry of each object, during the execution of data entry or spatial analysis operations. Others need to be specified by the user, in order to allow the system to store and maintain that information. OMT-G considers a set of five basic spatial relations between georeferenced classes, from which all others can be derived (Clementini, DiFelice & Oosterom, 1993): *touch, in, cross, overlap, and disjoint*. Since sometimes a larger set is required, due to cultural or semantic concepts that are familiar to the users, OMT-G also includes relations such as *adjacent to, coincide, contain, and near*, which are special cases of one of the five basic relations but deserve special treatment because of their common use in practice. Alternatively, the set of basic spatial relations derived from the 4-intersection matrix (*disjoint, contains/inside, covers/covered by, meet, overlap, equal*), or the spatial relations derived from the 9-intersection matrix, can be employed (Egenhofer & Franzosa, 1991; Egenhofer & Mark, 1995).

Considering these spatial relationship types, some spatial integrity rules can be established (Table 3). These rules are formulated using a notation commonly found in computational geometry, in which objects are indicated

Table 3. Spatial relationship integrity rules

Basic relations	
<b>Touch</b>	1. Let $A, B$ be two geo-objects, where neither $A$ nor $B$ are members of the <b>Point</b> class. Then $(A \text{ touch } B) = \text{TRUE} \Leftrightarrow (A^\circ \cap B^\circ = 0) \wedge (A \cap B \neq \emptyset)$ .
<b>In</b>	2. Let $A, B$ be two geo-objects. Then $(A \text{ in } B) = \text{TRUE} \Leftrightarrow (A \cap B = A) \wedge (A^\circ \cap B^\circ \neq \emptyset)$ .
<b>Cross</b>	3. Let $A$ be a geo-object of the <b>Line</b> class, and let $B$ be a geo-object of either the <b>Line</b> or the <b>Polygon</b> class. Then $(A \text{ cross } B) = \text{TRUE} \Leftrightarrow$ $\dim(A^\circ \cap B^\circ) = ((\max(\dim(A^\circ), \dim(B^\circ)) - 1) \wedge (A \cap B \neq A) \wedge (A \cap B \neq B))$
<b>Overlap</b>	4. Let $A, B$ be two geo-objects, both members of the <b>Line</b> or of the <b>Polygon</b> class. Then $(A \text{ overlap } B) = \text{TRUE} \Leftrightarrow$ $\dim(A^\circ) = \dim(B^\circ) = \dim(A^\circ \cap B^\circ) \wedge (A \cap B \neq A) \wedge (A \cap B \neq B)$ .
<b>Disjoint</b>	5. Let $A, B$ be two geo-objects. Then $(A \text{ disjoint } B) = \text{TRUE} \Leftrightarrow A \cap B = \emptyset$
Special cases	
<b>Adjacent to</b>	6. Let $A$ be a geo-object of the <b>Polygon</b> class and let $B$ be a geo-object of either the <b>Line</b> or the <b>Polygon</b> class. Then $(A \text{ adjacent to } B) = \text{TRUE} \Leftrightarrow (A \text{ touch } B) \wedge \dim(A \cap B) = 1$ .
<b>Coincide</b>	7. Let $A, B$ be two geo-objects. Then $(A \text{ coincide } B) = \text{TRUE} \Leftrightarrow A \cap B = A = B$ .
<b>Contain</b>	8. Let $A, B$ be two geo-objects, where $A$ is a member of the <b>Polygon</b> class. Then $(A \text{ contain } B) = \text{TRUE} \Leftrightarrow ((B \text{ in } A) = \text{TRUE}) \wedge ((A \text{ coincide } B) = \text{FALSE})$
<b>Near(<i>dist</i>)</b>	9. Let $A, B$ be two geo-objects. Let $C$ be a buffer, created at a distance <i>dist</i> around $A$ . Then $(A \text{ near}(\textit{dist}) B) = \text{TRUE} \Leftrightarrow (B \text{ disjoint } C) = \text{FALSE}$

Table 4. Spatial relationships

	Point	Line	Polygon
P o i n t			
S L i n e			
P o l y g o n			

by upper-case italic letters (e.g.,  $A, B$ ), their boundaries are denoted as  $\partial A$ , and their interiors as  $A^o$  (note that  $A^o = A - \partial A$ ). The boundary of a point object is considered to be always empty (therefore, the point is equivalent to its interior), and the boundary of a line is comprised of its two endpoints. A function, called *dim*, is used to return the dimension of an object and returns 0 if the object is a point, 1 if it is a line, or 2 if it is a polygon. Notice that additional constraints can be formulated in case some other relation is required by the application. Table 4 shows examples of the geometric interpretation of the spatial integrity rules between lines, points, and polygons.

Some relationships are only allowed between specific classes because they depend on the geometric representation. For instance, the existence of a *contain* relationship assumes that one of the classes involved is a polygon. In this aspect, traditional applications differ from geographic ones, where associations between conventional classes can be freely built.

The *near* rule is the only one described in Table 3 that requires a parameter. Since the notion of proximity varies according to the situation, a precise distance must be supplied in order to allow for the correct evaluation of the relationship.

In OMT-G, *network relationships* involve objects that are connected with each other. A network relationship only shows the need for a logical connection, not a requirement for the implementation of a particular structure. The connectivity rules, which apply to network relationship primitives, are listed in Table 5. The connec-

tion between all types of nodes and segments must be ensured by the system.

*Generalization* and *specialization* relationships do not generate the need for any additional integrity constraints, other than the ones expected in conventional databases; therefore, relationships of these kinds will not be treated here. *Aggregation* is a special form of association between objects, where one of them is considered to be assembled from others. The graphic notation used in OMT-G follows the one used by UML. An aggregation can occur between conventional classes, georeferenced classes, and georeferenced and conventional classes. When the aggregation is between georeferenced classes, *spatial aggregation* must be used (Figure 5). The usage of this kind of aggregation imposes spatial integrity constraints regarding the existence of the aggregated object and the corresponding sub-objects (Table 6): the geometry of each part is entirely contained within the geometry of the whole, and no overlapping among the parts is allowed. Also, the geometry of the whole is fully covered by the geometry of the parts. In Figure 5, city blocks are composed of parcels; that is, blocks are geometrically equivalent to the union of adjacent parcels. This implies that (1) no area belonging to the block can exist outside of a parcel, (2) no overlapping can occur among parcels that belong to a block, and (3) no area belonging to a parcel can exist outside of a block.

The spatial primitive *conceptual generalization* is used to record the need for different representations for a given object (Davis & Laender, 1999). In this type of

Table 5. Connectivity rules

<b>Arc-node structure</b>	Let $G = \{N, A\}$ be a network structure, composed of a set of nodes $N = \{n_0, n_1, \dots, n_p\}$ and a set of arcs $A = \{a_0, a_1, \dots, a_q\}$ . Members of $N$ and members of $A$ are related according to the following constraints: <ol style="list-style-type: none"> <li>For every node <math>n_i \in N</math> there must be at least one arc <math>a_k \in A</math>.</li> <li>For every arc <math>a_k \in A</math> there must be exactly two nodes <math>n_i, n_j \in N</math>.</li> </ol>
<b>Arc-arc structure</b>	Let $G = \{A\}$ be a network structure, composed of a set of arcs $A = \{a_0, a_1, \dots, a_q\}$ . Then the following constraint applies: <ol style="list-style-type: none"> <li>Every arc <math>a_k \in A</math> must be related to at least one other arc <math>a_i \in A</math>, where <math>k \neq i</math>.</li> </ol>

Figure 5. Spatial aggregation (“whole-part”)

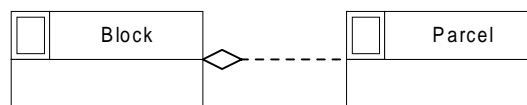


Table 6. Spatial aggregation integrity rules

<b>Spatial aggregation</b>	Let $P = \{P_0, P_1, \dots, P_n\}$ be a set of geo-objects. Then $P$ forms another object $W$ by spatial aggregation if and only if <ol style="list-style-type: none"> <li><math>P_i \cap W = P_i</math> for all <math>i</math> such that <math>0 \leq i \leq n</math>, and</li> <li><math>\left( W \cap \bigcup_{i=0}^n P_i \right) = W</math>, and</li> <li><math>((P_i \text{ touch } P_j) \vee (P_i \text{ disjoint } P_j)) = \text{TRUE}</math> for all <math>i, j</math> such that <math>i \neq j</math>.</li> </ol>
----------------------------	--

relationship, the superclass does not need to have a specific representation. However, its subclasses are allowed to inherit the superclass’ alphanumeric attributes while including their own representation and attributes. This primitive allows relationships involving specific representation styles to be made explicit. As previously shown, the way a class is represented influences the spatial relationship types that can occur. It is even possible to have the same representation alternative in more than one subclass because, in each one, the level of detail or resolution can be different. Conceptual generalization can occur in two representation variations: *according to geometric shape* and *according to scale* (Figure 6).

The variation according to geometric shape can also be used in the representation of classes that simultaneously have georeferenced and conventional instances. For instance, a traffic sign can exist in the database as a non-georeferenced object, such as a warehouse item, but it becomes georeferenced when installed at a particular location (Figure 7).

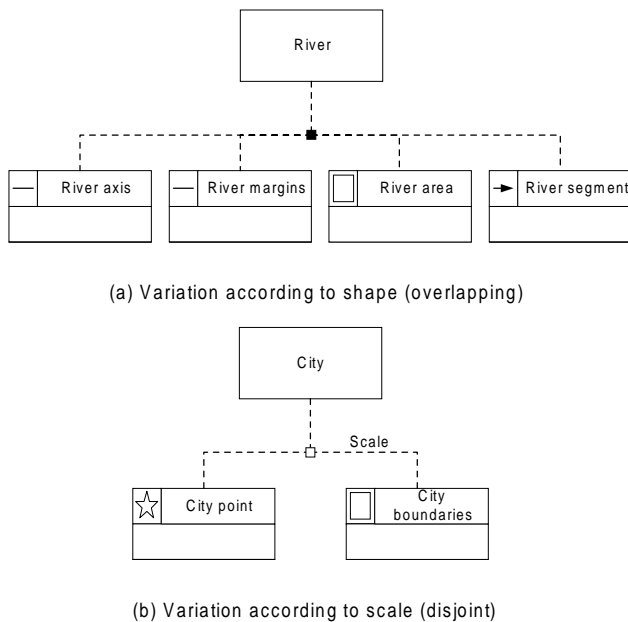
Except for the inheritance of superclass characteristics by subclasses, the conceptual generalization primitive does not define any additional spatial integrity constraints. However, subclasses are logically connected in

a way that can require the implementation of semantic integrity constraints. In Figure 6, for instance, city points and city boundaries must be coordinated to refer to the same real-world cities.

## FUTURE TRENDS

Current spatial DBMSs do not implement spatial integrity constraints in the same way they support relational integrity constraints. Rather, they assume that all the spatial integrity checking will be made by the GIS, during the data entry process and prior to the storage in the database. We feel that, if clear rules can be formulated, spatial integrity constraints can be translated from the conceptual schema to the implementation schema and could therefore be incorporated as a part of the spatial database’s design. A careful examination of the spatial integrity rules presented in this paper shows that every one of them can be implemented with the help of algorithms and data structures usually found in GIS products. The user must also be allowed to formulate specific spatial integrity constraints, as required by the application, in order to cover semantic

Figure 6. Conceptual generalization



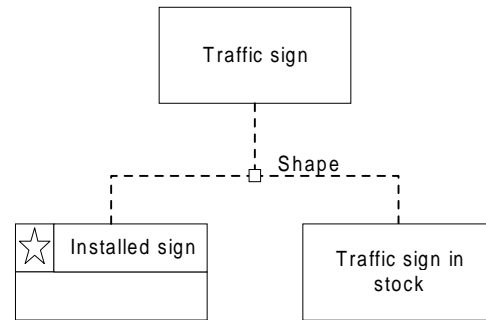
constraints. In our opinion, tools to create such integrity constraints will be available in next-generation spatial DBMSs.

## CONCLUSION

While constraint verifications can be incorporated to the GIS, one of the strongest arguments for installing a spatially-enabled DBMS in a corporate environment is to enable the use of a wide array of client products, each specializing in a specific aspect of the use of spatial data in the organization: database maintenance, spatial analysis, map production, integration with legacy systems, and so on. The best way to ensure that every modification of the spatial data results in an integral database is to implement the spatial integrity constraints as a function of the DBMS, adapting the client applications to reflect (and to benefit from) that functionality.

If spatial integrity constraints were included in spatially-enabled DBMSs, GIS developers and users would be able to invest on other aspects of the applications, such as multiple representations (Davis, 2000) and the use of ontologies to bring the application closer to the user's mental model (Fonseca, 2001; Fonseca, Davis & Câmara, 2003).

Figure 7. Conceptual generalization with a conventional class



## REFERENCES

- Borges, K.A.V., Davis, C.A., Jr., & Laender, A.H.F. (2001). OMT-G: An object-oriented data model for geographic applications. *GeoInformatica*, 5(3), 221-260.
- Borges, K.A.V., Davis, C.A., Jr., & Laender, A.H.F. (2002). Integrity constraints in spatial databases. In J.H. Doorn & L.C. Rivero (Eds.), *Database integrity: Challenges and solutions* (pp. 144-171). Hershey, PA: Idea Group Publishing.
- Câmara, G. (1995). *Models, languages, and architectures for geographic databases*. PhD thesis, INPE.
- Clementini, E., DiFelice, P., & Oosterom, P. (1993). A small set of formal topological relationships suitable for end-user interaction. *Proceedings of the 3rd Symposium on Spatial Database Systems* (pp. 277-295).
- Davis, C.A., Jr. (2000). *Multiple representations in geographic information systems*. PhD thesis, Universidade Federal de Minas Gerais, Belo Horizonte.
- Davis, C.A., Jr., & Laender, A.H.F. (1999). Multiple representations in GIS: Materialization through geometric, map generalization, and spatial analysis operations. *Proceedings of the 7th International Symposium on Advances in Geographic Information Systems (ACM GIS'99)* (pp. 60-65.)
- Egenhofer, M.J., & Franzosa, R.D. (1991). Point-set topological spatial relations. *International Journal of Geographical Information Systems*, 5(2), 161-174.
- Egenhofer, M.J., & Mark, D. (1995). Modeling conceptual neighborhoods of topological line-region relations. *International Journal of Geographical Information Systems*, 9(5), 555-565.



Elmasri, R., & Navathe, S. (2000). *Fundamentals of database systems* (3rd ed.). Reading, MA: Addison-Wesley.

Fonseca, F.T. (2001). *Ontology-driven geographic information systems*. PhD thesis, University of Maine.

Fonseca, F.T., Davis, C.A., Jr., & Câmara, G. (2003). Bridging ontologies and conceptual schemas in geographic applications development. *GeoInformatica*, 7(4), 355-378.

Frank, A.U., & Goodchild, M.F. (1990). *Two perspectives on geographical data modeling* (Tech. Rep. No. 90-11). National Center for Geographic Information and Analysis (NCGIA). Plumber, L., & Groger, G. (1997). Achieving integrity in geographic information systems: Maps and nested maps. *GeoInformatica*, 1(4), 346-367.

Rational Software Corporation. (1997, July). *The unified language: Notation guide*, version 1.1.

### KEY TERMS

**Geo-Fields:** Object classes that are used to represent objects and phenomena that are continuously distributed over the space. Examples of phenomena that are represented using geo-fields are temperature, rainfall, topography, soil type, and others.

**Geo-Objects:** Object classes that represent individual, particular real-world geographic objects. These objects can usually be traced back to individually identifiable elements, such as houses, lakes, and trees.

**GIS:** Geographic Information Systems are information systems that manage geographic data, that is, data for

which the geometric shape and spatial location are indispensable parts, which must be considered while working with them.

**Presentation:** In GIS, the set of properties of a geographic object that define its visual aspect, that is, the way the object is to be shown to the user. Involves aspects such as color, line type, line thickness, fill pattern, and others.

**Representation:** In GIS, a way to code the location, geometric shape, and topological behavior of objects. Involves the definition of aspects such as resolution, level of detail, spatial dimension.

**Semantic Integrity Constraints:** In GIS, constraints that are concerned with the meaning of geographic features. Semantic integrity constraints apply to database states that are valid by virtue of the properties of the objects that need to be stored.

**Spatial Integrity Constraints:** Rules that define a valid state of a spatial database. As with regular integrity constraints, spatial integrity constraints must be observed across updating operations in order to ensure that the state of the database is always valid. This includes rules on the valid behavior of the geometric and positional aspects of geographic data.

**Topology:** The branch of mathematics that deals with the properties of geometric configurations that are not altered by homomorphic transformations. In GIS, topology refers to a set of techniques that allows us to determine the relative positioning between two spatial objects, regardless of their exact coordinates.

# A Development Environment for Customer-Oriented Web Business

**Choongseok Lee**

*Korea Polytechnic University, Korea*

**Woojong Suh**

*Inha University, Korea*

**Heeseok Lee**

*Korea Advanced Institute of Science and Technology, Korea*

## INTRODUCTION

Web business can help companies strengthen the links between customers and suppliers. Under this environment, individual customers are getting much smarter and their needs are changing much faster than ever before. The customers can compare products and services with a variety of rich information provided from Web business systems, so that they can easily move to new products or services. Accordingly, many companies have conceived Web business systems to individual customers as a critical instrument for their business success and so have made a lot of efforts to develop and maintain them.

The implementation of Web business systems is complex and multidisciplinary. It requires several steps, such as information modeling for customers, navigation design for helping customers find information, user interface design for Web page layout, and actual implementation. To help develop Web business systems in a systematic fashion, many methodologies have been proposed (Lee, Suh, & Lee, 2004). The methodologies have their own advantages but are not powerful in overcoming the challenge of proper alignment of customers' needs with the business systems. From this perspective, recently, we provided a methodology to help develop and improve the customer-oriented Web applications (Lee et al.). However, looking at the speed of change in the Web business environment, it is becoming more difficult to employ the methodology without an automated support environment. Therefore, this article presents an environment, called *eBizBench*, for the effective development and maintenance of customer-oriented Web business systems.

This article first discusses the previous development environments for Web business systems in a background section. Then as a main trust of this article, the architecture of the *eBizBench* along with its screen examples is presented, and the *eBizBench* is compared with other

development environments. Finally, this article discusses future trends and makes a conclusion.

## BACKGROUND

Currently, in constructing Web business systems developers tend to pay little attention to requirements elicitation, requirements analysis, or reusability (Taylor, McWilliam, Forsyth, & Wade, 2002). They may use ad hoc tools that depend on their expertise or experience. However, this ad hoc development of Web business system without any rigorous development environment may cause Web crises such as delayed schedule, exceeded budget, and poor quality (Ginige & Murugesan, 2001).

To help develop Web business systems in a robust and coherent fashion, several development environments have been proposed. These development environments can strengthen the quality of the resulting Web business. They can be categorized into two types: implementation-oriented or integrated. The implementation-oriented environments focus on the generation of database and Web pages rather than conceptual design. They include WebDesigner (Bichler & Nusser, 1996), W3Objects (Ingham, Caughey, & Little, 1997), STRUDEL (Levy, Florescu, Suciu, Kang, & Fernandez, 1997), Jessica (Barta & Schranz, 1998), WebComposition Markup Language (WCML; Gaedke, Schempf, & Gellersen, 1999), ReWeb and TestWeb (Ricca & Tonella, 2001), and MyXML (Kirda, Jazayeri, & Kerer, 2001). The integrated environments cover phases ranging from conceptual design to implementation details. OOHDM-Web (Schwabe, Pontes, & Moura, 1999), Araneus (Mecca, Merialdo, Atzeni, & Crescenzi, 1999), AutoWeb (Fraternali & Paolini, 2000), JWeb (Garzotto, Paolini, & Baresi, 2001), and OO-H CASE Tool (Gómez & Cachero, 2001) belong to this integrated category. The

implementation-oriented development environments tend to evolve into integrated ones.

On the other hand, in the Web business environment, the speed of change is inconceivable; companies should be able to analyze customers' experiences and respond to their needs agilely. For obtaining these capabilities, Web business systems also need to be developed and evolved rapidly. Yet, the previous system-development environment for Web business systems falls back on technical details such as conceptual design and visual code generation. Especially, they offer little opportunity for the kind of customer analysis over time. Accordingly, they have a limit on effectively fitting technical details into the real lives of the customers. Furthermore, the development environments can leap in with more reuses of analysis results. Getting work done nowadays depends on getting designers to share their design results. For the above customer orientation and reusability, the eBizBench was developed as an integrated development environment.

### eBizBench ARCHITECTURE

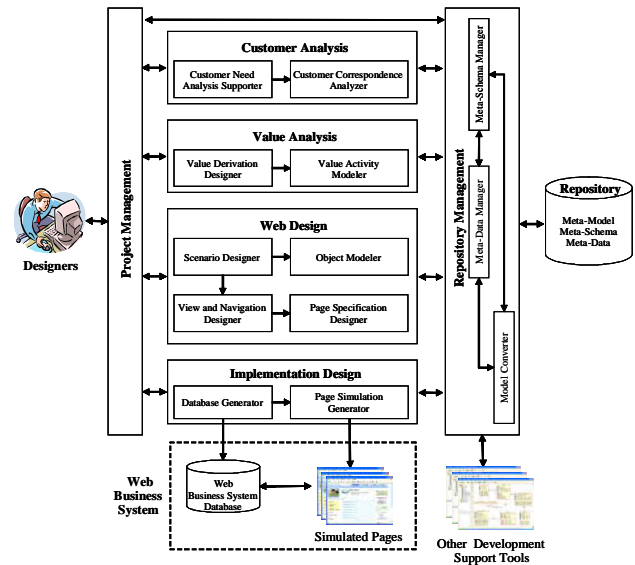
The eBizBench supports whole activities for development in a seamless fashion, ranging from customer needs analysis to system implementation. As the development of Web business systems becomes complicated, it becomes almost impossible to build the system manually; it is important to catapult the Internet business to high quality when development time and money are scarce. An automated environment for supporting development activities is desired. The proposed eBizBench employs a repository as the basis of the reuse and conversion of design results.

The eBizBench has been implemented using Microsoft Visual Basic 6.0, Active Server Page (ASP), and Microsoft SQL Server 7.0 database management system (DBMS) in the Windows environment. The DBMS manages the repository. The overall architecture of the eBizBench is depicted in Figure 1. The eBizBench consists of seven subsystems: (i) project management, (ii) customer analysis, (iii) value analysis, (iv) Web design, (v) implementation design, (vi) repository management, and (vii) repository.

The project management subsystem interacts with the other subsystems. It can manage a variety of system development projects.

The customer analysis subsystem consists of the customer need analysis supporter and customer correspondence analyzer. Customers are categorized into customer groups according to their features. Each customer group has different needs. Customer groups and

Figure 1. Architecture of the eBizBench



the corresponding needs are summarized in the customer need analysis supporter, and then their correspondence is analyzed by the use of the customer correspondence analyzer, as shown in Figure 2.

The objective of the value analysis is to model value activities. This subsystem consists of the value derivation designer and value activity modeler. In the value derivation designer, customers' needs are analyzed and then prioritized in the form of the value derivation table. The value activity modeler identifies customers' requests in the form of events, as shown in Figure 3.

The Web design subsystem deals with functionalities for the conceptual design. It includes the scenario designer, object modeler, view and navigation designer, and page specification designer. The scenario designer

Figure 2. A screen example of the customer correspondence analyzer

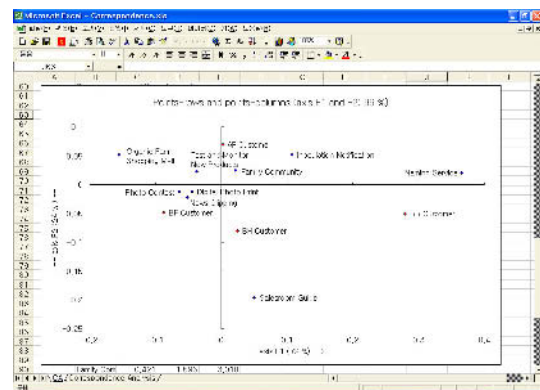


Figure 3. A screen example of the value activity modeler

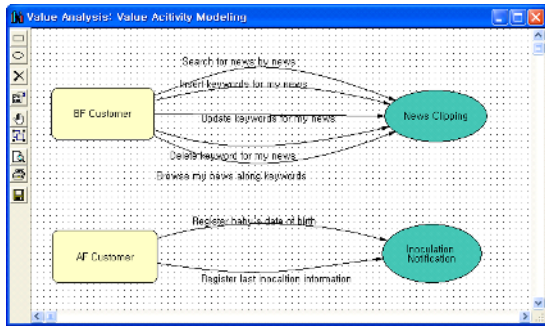


Figure 4. A screen example of the object modeler

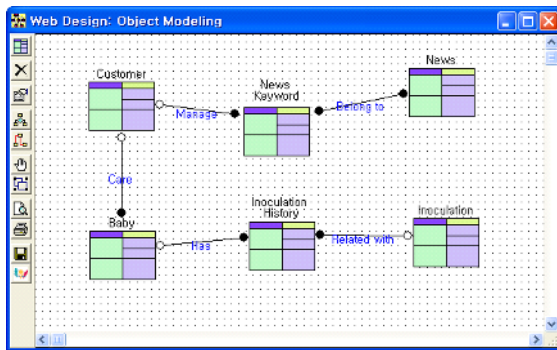
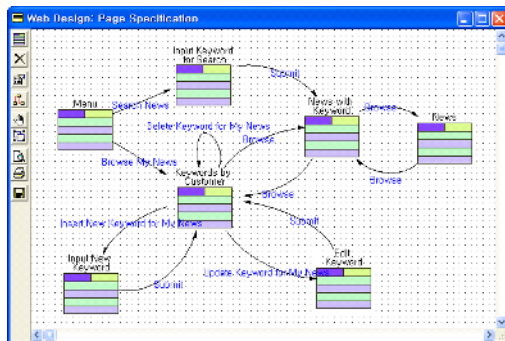


Figure 5. A screen example of the page specification designer



describes scenarios in the form of natural language. These scenarios lead to an object model in the object modeler, as shown in Figure 4. In the view and navigation designer, information units that customers want to find are designed as object-oriented views, and then navigational paths for customers to access information are designed. Finally, through the page specification designer, users' informa-

tion windows and the flow from one page to another are defined according to the views and their navigational paths, as shown in Figure 5.

The implementation design subsystem consists of the database generator and page simulation generator. The object model and object-oriented views are transformed into logical database schema, and then physical databases are generated for the target DBMS by using the database generator. The page simulation generator provides simulated navigation among pages.

The repository management subsystem not only stores and retrieves the design results but also converts them for other development support tools. It consists of the meta-schema manager, meta-data manager, and model converter. The repository consists of three layers: meta-model, meta-schema, and meta-data. The meta-model stores the meta-information about design models and their relationships. The instance of meta-model is the meta-schema. The meta-data includes the design results for a specific Web business system and the instance of the meta-schema. The functions of each subsystem are summarized in Table 1.

## COMPARISONS OF DEVELOPMENT ENVIRONMENTS

Several development environments for developing Web business systems have been proposed. These environments may include OOHDM-Web (Schwabe et al., 1999), Araneus (Mecca et al., 1999), AutoWeb (Fraternali & Paolini, 2000), JWeb (Garzotto et al., 2001), and OO-H CASE Tool (Gómez & Cachero, 2001). The OOHDM-Web is a development environment for the OOHDM (Schwabe & Rossi, 1995). It implements templates (a mixture of HTML) and calls for the functions (in library) that can provide access to the navigational objects stored in a relational database. The Araneus focuses on the management of the unstructured and structured Web contents. The AutoWeb adopts a model-driven development approach. The JWeb supports HDM-driven design of hypermedia applications. OO-H CASE Tool supports the OO-H method. The emphasis of the OO-H method is on capturing all relevant information to create device-independent front-end specifications.

The comparison of these development environments is summarized in Table 2. The customer analysis is a starting point for Web business system development. Ease of use and customer satisfaction are important. The emphasis of our eBizBench is on aligning customers' needs with implementation details. The eBizBench provides the customer need analysis supporter module



*Table 1. A summary of the eBizBench functions*

Subsystem	Module	Function
■ Project Management	■ Project Manager	<ul style="list-style-type: none"> <li>■ Retrieve previous Internet business system projects</li> <li>■ Update a previous Internet business system project</li> <li>■ Create a new Internet business system project</li> </ul>
■ Customer Analysis	■ Customer Need Analysis Supporter	■ Document customer groups and their needs
	■ Customer Correspondence Analyzer	■ Analyze correspondences among customer groups and their needs
■ Value Analysis	■ Value Derivation Designer	<ul style="list-style-type: none"> <li>■ Identify customers' values</li> <li>■ Derive value activities to achieve customers' values</li> <li>■ Determine implementation priority for customer needs</li> </ul>
	■ Value Activity Modeler	■ Specify events between customer groups and Internet business system
■ Web Design	■ Scenario Designer	■ Describe scenarios for each event
	■ Object Modeler	■ Design objects and relationships among objects
	■ View and Navigation Designer	<ul style="list-style-type: none"> <li>■ Define object-oriented (OO) views</li> <li>■ Define access structure nodes (ASNs)</li> <li>■ Specify navigational links among OO views and ASNs</li> </ul>
	■ Page Specification Designer	<ul style="list-style-type: none"> <li>■ Specify structures for pages</li> <li>■ Define navigation links among pages</li> </ul>
■ Implementation Design	■ Database Generator	■ Generate physical database schema for target DBMS
	■ Page Simulation Generator	■ Generate simulation pages
■ Repository Management	■ Model Converter	■ Convert a model instance into another model
	■ Meta-Data Manager	<ul style="list-style-type: none"> <li>■ Manage instances for meta-schema</li> <li>■ Browse meta-data for specific project</li> </ul>
	■ Meta-Schema Manager	■ Manage instances for meta-model
■ Repository	■ Meta-Model	■ Store meta-information about model
	■ Meta-Schema	■ Store meta-information about model components
	■ Meta-Data	■ Store meta-information about specific Internet business systems

and customer correspondence analyzer module for the systematic analysis of customer needs. The eBizBench adopts scenarios that can describe processes between customers and the Internet business system. The use of scenarios is likely to enhance the usability of the Web business system by reflecting customers' navigational requirements effectively. It can strike the balance between customers' needs and conceptual design.

The conceptual design prior to physical implementation is essential. Internet business design is not an exception. The conceptual design of Internet business systems includes information modeling, navigation design, and user interface design. The Araneus uses ER design for information modeling and Araneus Data Model (ADM) for navigation design. The JWeb is based on the hypermedia design model (HDM; Garzotto, Mainetti, & Paolini, 1995), which is based on ER and consists of structure design, navigation design, and presentation design. The AutoWeb uses HDM-Lite, an adaptation of HDM especially tailored for Web business systems. By contrast, the OOHDM-Web uses a conceptual model, navigation model, and interface model according to an object-oriented methodology called OOHDM (Schwabe & Rossi, 1995). The OO-H CASE Tool provides a class diagram, navigational access diagram (NAD), and abstract presentation diagram (APD). Similarly, the eBizBench uses an object model; the object orientation is better for reusability.

Because Web business systems continue to evolve and improve according to customers' needs, their devel-

opment is not a one-time event, but a process with a long life cycle. For this purpose, the Araneus and AutoWeb employ design repository and meta-data. Furthermore, our eBizBench includes the repository and its management modules for the reuse and conversion of design results among heterogeneous development environments.

Several systems generate CGI scripts, page templates, and database tables. The eBizBench generates not only database tables but also simulated pages. By the use of these simulated pages, usability of Internet business systems can be enhanced. The Araneus, AutoWeb, and JWeb are developed by the use of Sun Microsystems' Java for portability on different platforms. Currently, the eBizBench is optimized for the Microsoft Windows platforms; we are in the process of enhancing the portability of the eBizBench.

## FUTURE TRENDS

Seeing the evolution of research trends, the following research directions may be the matters of concern. The first concern is to incorporate the usability metrics to strengthen the feasibility of Web business systems into a development environment. The advantages of this automation have been pointed out as cost reduction, consistency, and coverage (Ivory & Hearst, 2001). Although many metrics evaluating Web business systems have been proposed, it is still an important challenge to



Table 2. Comparisons of development environments

System Criteria	OOHDM-Web	Araueus	AutoWeb	JWeb	OO-H CASE Tool	eBizBench
Reference	(Schwabe et al., 1999)	(Mecca et al., 1999)	(Fraternali & Paolini, 2000)	(Garzotto et al., 2001)	(Gómez & Cucho, 2001)	This article
Customer Analysis	N/A	N/A	N/A	N/A	N/A	- Customer Need Analysis - Customer Correspondence Analysis
System Requirements Analysis	N/A	N/A	N/A	N/A	N/A	Scenario
Data Model	OO	ER	ER	ER	OO	OO
Information Modeling	- Conceptual Model	- ER Design	- Structure Design	- Structure Design	- Class Diagram	- Object Modeling
Navigation Design	- Navigation View	- ADM Design	- Navigation Design	- Navigation Design	- NAD	- View Design - Navigation Design
User Interface Design	- Interface Model	N/A	- Presentation Design	- Presentation Design	- APD	- Page Schema Design
Implementation Support	- CGI	- Database - Web Page Template	- Database - Web Page	- Database - Web Page Template	- Page Template	- Database - Page Template - Simulation of Navigation
Meta-data	N/A	Design Repository	Meta-data	HDM-Schema	N/A	Repository
Main Purpose	Template-driven Design	Design and Development	Model-driven Design, Implementation, and Maintenance	Design, Reuse, Documentation, and Tools Communication	Device Independent Design and Implementation	Customer Requirements Analysis, Design, Implementation, and Reuse
Development tool	CGI Scripts	Java	Java	Java	Not Specified	Visual Basic 6.0 and ASP
Portability	Not Portable	Portable	Portable	Portable	N/A	Not Portable
Components	- Authoring Environments - Browsing Environments - Maintenance Environments	- Ulixes Editor - Minerva - ADM Object Manager - Penelope - Design Repository	- Diagram Editor - Relational Schema Generator - Prototype Generator - Style Sheet Editor - Page Generator - Data Entry Generator - Page Grabber - Administrator - Meta-Data	- Schema Editor - Mapper - Instance Editor - Configurator - Generator - HDM-Schema	- Editor - Model Compiler	- Project Manager - Customer Need Analysis Supporter - Customer Correspondence Analyzer - Value Derivation Designer - Value Activity Modeler - Scenario Designer - Object Modeler - View and Navigation Designer - Page Specification Designer - Database Generator - Page Simulation Generator - Menu-Schema Manager - Menu-Data Manager - Model Converter

develop the metrics theoretically and empirically well defined (Calero, Ruiz, & Piattini, 2004).

The second concern is to employ Web mining to support customer analysis of the development environment. Until recently, it has been attempted to adopt Web mining to guide the design and the continuous management of customer-oriented Web business systems (Albert, Goes, & Gupta, 2004). The adoption of data

mining can extract potentially useful patterns of customers through tracking the customers and their behaviors on the Web (Spiliopoulou, 2000). These patterns can help developers improve Web business systems to reflect customers' needs on a real-time basis. Accordingly, to incorporate data mining methods into the customer analysis mechanisms in a development environment systematically will be another challenge.

## CONCLUSION

This article proposes a development environment, eBizBench, for implementing Web business systems. It has two features different from those of the existing development environments. First, its emphasis is on systematic support for customer-oriented development. Customer needs are analyzed at an early stage. By using the eBizBench, companies are more likely to capture the context in which customers interact with their Internet business system. Second, a repository and the corresponding repository management subsystem are implemented for the reuse and conversion of design results. They are better able to enhance the productivity of development process as well as the maintenance of the resulting system. Nevertheless, the proposed environment will be improved by solving the challenges discussed in the above future trends.

## REFERENCES

Albert, T. C., Goes, P. B., & Gupta, A. (2004). GIST: A model for design and management of content and interactivity of customer-centric Web sites. *MIS Quarterly*, 28(2), 161-182.

Barta, R. A., & Schranz, M. W. (1998). JESSICA: An object-oriented hypermedia publishing processor. *Computer Networks and ISDN Systems*, 30, 281-290.

Bichler, M., & Nusser, S. (1996). Modular design of complex Web-applications with W3DT. *Proceedings of the Fifth Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (pp. 328-333).

Calero, C., Ruiz, J., & Piattini, M. (2004). A Web metrics survey using WQM. *International Conference on Web Engineering* (pp. 147-160).

Fraternali, P., & Paolini, P. (2000). Model-driven development of Web applications: The AutoWeb system. *ACM Transactions on Information Systems*, 28(4), 323-382.

Gaedke, M., Schempf, D., & Gellersen, H.-W. (1999). WCML: An enabling technology for the reuse in object-oriented Web engineering, *Poster-Proceedings of the Eighth International World Wide Web Conference* (pp. 102-103).

Garzotto, F., Mainetti, L., & Paolini, P. (1995). Hypermedia design, analysis, and evaluation issues. *Communications of the ACM*, 38(8), 74-86.

Garzotto, F., Paolini, P., & Baresi, L. (2001). Supporting reusable Web design with HDM-Edit. *Proceedings*

*of the 34th Hawaii International Conference on System Sciences* (p. 7).

Ginige, A., & Murugesan, S. (2001). Web engineering: An introduction. *IEEE Multimedia*, 8(1), 14-18.

Gómez, J., & Cachero, C. (2001). Conceptual modeling of device-independent Web applications. *IEEE Multimedia*, 8(2), 26-39.

Ingham, D. B., Caughey, S. J., & Little, M. C. (1997). Supporting highly manageable Web services. *Computer Networks and ISDN Systems*, 29, 1405-1416.

Ivory, M. Y., & Hearst, M. A. (2001). The state of the art in automating usability evaluation of user interfaces. *ACM Computing Surveys*, 33(4), 470-516.

Kirda, E., Jazayeri, M., & Kerer, C. (2001). Experiences in engineering flexible Web services. *IEEE Multimedia*, 8(1), 58-65.

Lee, C., Suh, W., & Lee, H. (2004). Implementing a community Web site: A scenario-based methodology. *Information and Software Technology*, 46(1), 17-33.

Levy, A., Florescu, D., Suciu, D., Kang, J., & Fernandez, M. (1997). STRUDEL: A Web-site management system. *Proceedings ACM SIGMOD International Conference on Management of Data* (Vol. 26, No. 2, pp. 549-552).

Mecca, G., Merialdo, P., Atzeni, P., & Crescenzi, V. (1999). The (short) Araneus guide to Web-site development. *Proceedings of the second International Workshop on the Web and Databases in conjunction with SIGMOD* (pp. 167-177).

Ricca, F., & Tonella, P. (2001). Understanding and restructuring Web sites with ReWeb. *IEEE Multimedia*, 8(2), 40-51.

Schwabe, D., Pontes, R. A., & Moura, I. (1999). OOHDM-Web: An environment for implementation of hypermedia applications in the WWW. *SigWEB Newsletter*, 8(2).

Schwabe, D., & Rossi, G. (1995). The object-oriented hypermedia design model. *Communications of the ACM*, 38(8), 45-46.

Spiliopoulou, M. (2000). Web usage mining for Web site evaluation. *Communications of the ACM*, 43(8), 127-134.

Taylor, M. J., McWilliam, J., Forsyth, H., & Wade, S. (2002). Methodologies and Website development: A survey of practice. *Information and Software Technology*, 44(6), 381-391.

## **KEY TERMS**

**ASP:** An Active Server Page (ASP) is an HTML page that includes one or more scripts that are processed on a Microsoft Web server before the page is sent to the user. An ASP is somewhat similar to a server-side include or a common gateway interface (CGI) application in that all involve programs that run on the server, usually tailoring a page for the user.

**CASE:** Computer-Aided Software (or Systems) Engineering. Software tools that provide computer-assisted support for some portion of the software or systems development process, especially on large and complex projects involving many software components and people.

**HTML:** Hypertext Markup Language. It is a markup language, using tags in pairs of angle brackets, for identifying and representing the Web structure and layout through Web browsers.

**Meta-Data:** Data that describe the properties or characteristics of other data, including the structure, source, use, value, and meaning of the data. It is often described as data of data, briefly.

**Repository:** A repository is a centralized database where meta-data about database structure, applications, Web pages, users, and other application components is stored and maintained. It provides a set of mechanisms and structures to achieve seamless data-to-tool and data-to-data integration.

**Scenario:** A scenario is similar to a use case or script that describes interactions at a technical level. Scenarios are more informal and capture customers' requirements in a natural fashion.

**Web Mining:** Web mining is the integration of information gathered by traditional data mining methodologies and techniques with information gathered over the World Wide Web. Web mining is used to capture customer behavior, evaluate the effectiveness of a particular Web site, and help quantify the success of a marketing campaign.

# Digital Media Warehouses

**Menzo Windhouwer**

*Center for Mathematics and Computer Science, The Netherlands*

**Martin Kersten**

*Center for Mathematics and Computer Science, The Netherlands*

## INTRODUCTION

Due to global trends, like the rise of the Internet, the cheapness of storage space and the ease of digital media acquisition, vast collections of digital media, are becoming ubiquitous. Futuristic usage scenarios, like ambient technologies, strive to open up these collections for the consumer market. However, this requires high-level semantic knowledge of the items in the collections. Experiences in the development and usage of multimedia retrieval systems have shown that within a specific, but still limited domain, semantic knowledge can be automatically extracted and exploited. However, when the domain is unspecified and unrestricted, that is, the collection becomes a warehouse, semantic knowledge quickly boils down to generics. The research on *Digital Media Warehouses* (DMWS) focuses on improving this situation by providing additional support for annotation extraction and maintenance to build semantically rich knowledge bases.

The following sections will introduce these DMW topics for closely related topics like multimedia storage, the usage/indexing of the extracted annotations for multimedia retrieval, and so forth, the reader is directed to the wide variety of literature on multimedia databases (for example, Subrahmaniam, 1997).

## BACKGROUND

Media objects are semi or even unstructured by nature. They are mostly sensor data, a natural phenomenon represented in a series of numbers. There is no, or very limited, meta-data available. Adding annotations is a labor-intensive work, which has been for a long time already the task of librarians. However, the vastness and diversity of DMWS makes this approach no longer feasible. Automatic annotation programs are envisioned to take over or support this task.

The automatic extraction of semantically meaningful annotations (also known as features) has been, and still is, a major research area in computer vision. A main problem is bridging the semantic gap. For visual data, this gap can be defined as follows:

*The semantic gap is the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data has for a user in a given situation* (Smeulders et al., 2000, p. 1349).

This definition can be generalized to digital media objects of any modality without loss of validity.

The semantic gap is not bridged and may never be. One of the main reasons is the role of ambiguity. The more abstract a semantic concept is, the more subjective interpretations are possible, for example, due to cultural context-sensitivity. Eakins (1996) distinguishes three levels of features:

- **Level 1:** primitive features; for example, color, texture and shape;
- **Level 2:** derived (or logical) features; for example, containment objects of a given type or individual objects;
- **Level 3:** abstract attributes; for example, named events or types of activities, or emotional or religious significance.

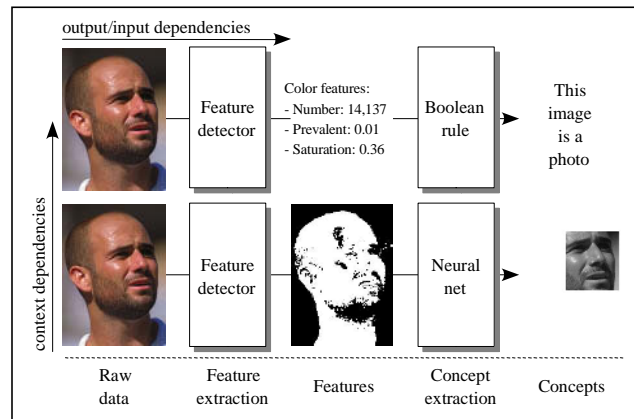
The higher the level the more subjective, and thus ambiguous, annotations become. State of the art annotation extraction algorithms reach level 2. Level 3 algorithms are currently only applicable in clearly defined and distinguishable (narrow) domains.

The most popular approach is to start with level 1 features and aggregate those into higher level semantically meaningful concepts. Common intuition is that annotations of multiple modalities should be combined, hopefully improving and enforcing each other's findings.

A wide variety of extraction algorithms can be found in scientific publications, mostly dedicated to one specific modality (Del Bimbo, 1999; Foote, 1999; Jurafski & Martin, 2000). Models to combine features range from hard-coded rules to machine learning algorithms (Mitchell, 1997). Examples of frameworks encompassing these techniques are COBRA (Petkovic, 2003) and the *compositional semantics* method used in Colombo, Del Bimbo, and Pala (1999).

Figure 1 shows an annotation example, which uses various extraction algorithms: detectors to extract basic

Figure 1. Example of annotation extraction and dependencies



(level 1) features (color features and a skin map), a hand-coded boolean rule to classify the image as a photograph and a neural network, to determine if the photo contains a face.

## ANNOTATION EXTRACTION

Higher level annotations depend on the availability of lower level annotations. These dependencies imply an order of execution of extraction algorithms. A fixed set of algorithms can never take all possible modalities and contexts into account and produce all needed subjective and objective annotations. So, the order will not be hard-coded but described in a declarative specification, thus allowing easy extensibility.

The annotation extraction process is concerned with executing the correct extraction algorithms in the specified order. Two basic implementation approaches are:

1. **The pool approach:** Annotations live in a data pool. Extraction algorithms become active when their input is available. This approach, where the algorithms are known as *daemons*, is taken in the LIBRARY extension to the ALTA VISTA indexing engine (De Vries, Eberman & Kovalcin, 1998).
2. **The pipeline approach:** Annotations are pushed or pulled through a pipeline of extraction algorithms. The *processing graphs* of the MOODS system (Griffioen, Yavatkar & Adams, 1997) follow this approach.

When ambiguity is allowed, special care should be taken to maintain the complete context of the annotations in the extraction process. This context will serve as a means for supporting disambiguation at query time. One approach is to store the annotations in a tree, where a path from the root to an annotation reflects the processing

order. Whole subtrees can thus be easily culled out. This approach is taken in the ACOI system (Windhouwer, 2003), where non-deterministic context-sensitive grammars are used to describe the processing order and (ambiguous) annotations live in the resulting parse forests. This system distinguishes two types of dependencies (also shown in Figure 1):

1. **Output/input dependencies:** An algorithm depends on the annotations produced by another algorithm; for example, the face detector depends on the skin detector.
2. **Context dependencies:** An algorithm is deliberately placed in the context of another algorithm; for example, the face detector is only executed when the photo detector was successful.

While output/input dependencies can be automatically discovered based on the algorithms signatures (Haidar, Joly & Bahsoun, 2004), the context dependencies are domain-based design decisions explicitly made by the developer of the DMW.

## ANNOTATION MAINTENANCE

The approaches outlined in the previous section are able to produce the collection of annotations, as far as they can be (semi-) automatically extracted, to describe the semantic content of a media object (in its context). However, in due time new extraction algorithms will become available, making it possible to capture a new (subjective and ambiguous) viewpoint or improve old ones. Also the source media objects may change over time. This may happen beyond the control of the DMW, as it may be a virtual collection (the objects do not reside in and are not controlled by the DMW).



All these (external) sources of change lead to start a new extraction process for (all) media objects. However, extraction algorithms can be very expensive: a frame-by-frame analysis of a video may take several hours. Reruns of these expensive algorithms should be prevented. The dependencies of a new or updated algorithm should thus be investigated to localize affected annotations and limited the number of revalidation runs. Support for this kind of incremental maintenance of media annotations is still in its infancy. The ACOI system (Windhouwer, 2003) supports this by deriving a dependency graph of the process specification. This graph is analyzed, and an incremental extraction process is started which will only produce the affected annotations.

Looking once more at Figure 1, the constants in the photo decision rule can be changed, that is, not affecting the output/input dependencies without re-executing the lower level feature extraction. However, after revalidation of the updated decision rule, it may be needed to also revalidate the face detector; that is, the dependencies are followed to incrementally update the annotations in the DMW.

## FUTURE TRENDS

As mentioned in the introduction, envisioned application scenarios, like ambient environments, contribute to an even more rapid growth of DMWs. So, the future will show an even greater need for transparent management of the annotation process. Consumers will only spend a limited amount of time for manual annotation or supervision of a learning process, which will stress the need for incremental maintenance with its capability to leverage the reuse of already accumulated knowledge. The manual annotation burden can further be decreased when MPEG-7 (Martínez, 2003) becomes more common. Commercial digital media will then, by default, provide more detailed meta-data, which may provide a good starting point for more advanced and personal annotations.

The two basic approaches to steering annotation extraction; that is, the pool and pipeline approach are based on very basic computer science models. Other (research) areas where these models are common; for example, workflow management and database view maintenance can provide more insight for their proper and advanced use in this new area. For example, a coordination system like Galaxy (Galaxy, 2003) closely resembles an annotation extraction management system, and study of its underlying hub model may provide fruitful insights for more dynamic annotation systems.

## CONCLUSION

Multimedia collections are growing and diverging at a rapid rate, turning into DMWs. However, opening up these type of collections poses a major problem. Manual annotating the whole collection is too labor intensive and will have to be supported by annotation extraction algorithms. As the set of needed algorithms and annotations will dynamically grow over time, they have to be managed in a transparent way. Executing a declarative specification of this process is already well understood and bears many resemblances to existing practices. However, incremental maintenance of the created collection of annotations is still in its infancy, but bearing future trends in mind will be a key issue in building useful and maintainable collection.

## REFERENCES

- Colombo, C., Del Bimbo, A., & Pala, P. (1999). Semantics in visual information retrieval. *IEEE MultiMedia*, 6(3), 38-53.
- De Vries, A.P., Eberman, B., & Kovalcin, D.E. (1998). The design and implementation of an infrastructure for multimedia digital libraries. *Proceedings of the 1998 International Database Engineering & Applications Symposium* (pp. 103-110).
- Del Bimbo, A. (1999). *Visual information retrieval*. San Francisco: Morgan Kaufmann.
- Eakins, J.P. (1996). Automatic image content retrieval: Are we getting anywhere? *Proceedings of the 3rd International Conference in Electronic Library and Visual Information Research* (pp.123-135).
- Foote, J. (1999). An overview of audio information retrieval. *Multimedia Systems*, 7(1), 2-10.
- Galaxy. (2003). Galaxy Communicator. Retrieved February 3, 2005, from <http://communicator.sourceforge.net/>
- Griffioen, J., Yavatkar, R., & Adams, R. (1997). A framework for developing content-based retrieval systems. *Intelligent multimedia retrieval*. Cambridge, MA: AAAI Press/MIT Press.
- Haidar, B., Joly, P., & Bahsoun, J.-P. (2004, Month 00). An open platform for dynamic multimedia indexing. *Proceedings of the 5th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2004)*, Lisboa, Portugal.

Jurafsky, D., & Martin, J.H. (2000). *Speech and language processing*. Prentice Hall.

Martínez, J.M. (2003). MPEG-7 overview. Retrieved February 3, 2005, from <http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm>

Mitchell, T. (1997). *Machine learning*. McGraw-Hill.

Petkovic, M. (2003). *Content-based video retrieval supported by database technology*. PhD thesis, Centre for Telematics and Information Technology, Enschede, The Netherlands.

Smeulders, A.W., Worring, M., Santini, S., Gupta, A., & Jain, R. (2000). Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12), 1349-1380.

Subrahmaniam, V.S. (1997). *Principles of multimedia database systems*. San Francisco: Morgan Kaufmann.

Windhouwer, M.A. (2003). *Feature grammar systems: Incremental maintenance of indexes to digital media warehouses*. PhD thesis, University of Amsterdam, Amsterdam, The Netherlands.

## KEY TERMS

**Annotation:** Information about an (multimedia) object. This information may directly describe the (semantic) content of an object (e.g., this photo shows a cityscape) or describe its relations to other (external) objects (for example, this photo was made in the year 1999).

**Annotation Extraction Algorithm:** An algorithm that automatically extracts (set of) annotations, which describe (the content of) a media object. The input of such an algorithm can consist of the media object itself combined with previously extracted annotations or other additional information.

**Annotation Extraction Process:** The (semi-) automatic extraction of annotations aimed at describing (the content of) a digital media object. In this process, annotation extraction algorithms are called in the correct order, so their out- and input dependencies are met.

**Annotation Maintenance:** When the annotation extraction algorithms or the media objects change, the already extracted annotations have to be revalidated. If a dependency description for the extraction algorithms is available, an incremental extraction process can be started, where only the affected annotations are (re)produced.

**Annotation Pipeline:** A media object (and its growing collection of annotations) is pushed or pulled through a sequence of annotation extraction algorithms.

**Annotation Pool:** Annotations of digital media are stored in a data pool. Annotation extraction algorithms populate this pool when their input is available.

**Digital Media Warehouse:** A vast collection of digitized media objects from an unrestricted set of different domains.

**Semantic Gap:** The lack of coincidence between the information that one can extract from the data and the interpretation that the same data has for a user in a given situation.

# Discovering Association Rules in Temporal Databases

**Juan M. Ale**

*Universidad de Buenos Aires, Argentina*

**Gustavo H. Rossi**

*Universidad Nacional de La Plata, Argentina*

## INTRODUCTION

The problem of the discovery of association rules comes from the need to discover interesting patterns in transaction data in a supermarket. Since transaction data are temporal we expect to find patterns that depend on time. For example, when gathering data about products purchased in a supermarket, the time of the purchase is stamped in the transaction.

In large data volumes, as used for data mining purposes, we may find information related to products that did not necessarily exist throughout the complete data-gathering period. So we can find a new product, such as a DVD player, that was introduced after the beginning of the gathering, as well as a product, like a 5 1/4-inch flexible disk unit, that had been discontinued before the ending of the same gathering. It would be possible that that new product could participate in the associations, but it may not be included in any rule because of support restrictions. Suppose we have gathered transactions during 10 years. If the total number of transactions is 10,000,000 and we fix as minimum support 0.5%, then a particular product must appear in, at least, 50,000 transactions to be considered frequent. Now, take a product that has been sold during these 10 years and has just the minimum support: It appears on average in 5,000 transactions per year. Consider now another product that was incorporated two years ago and that appears in 20,000 transactions per year. The total number of transactions in which it occurs is 40,000; for that reason, it is not frequent, even though it is four times as popular as the first one. However, if we consider just the transactions generated since the product appeared in the market, its support might be above the stipulated minimum. In our example, the support for the new product would be 2%, relative to its lifetime, since in two years the total of transactions would be about 2,000,000 and this product appears in 40,000 of them. Therefore, these new products should appear in interesting and potentially useful association rules. Moreover, we should consider the case of some products that may be frequent just in some subintervals strictly contained in their period of life but not in the entire interval corresponding to their lifespan.

We solve this problem by incorporating time in the model of discovery of association rules. We call these new rules *general temporal association rules*.

One by-product of this idea is the possibility of eliminating outdated rules, according to the user's criteria. Moreover, it is possible to delete obsolete sets of items as a function of their lifetime, reducing the amount of work to be done in the determination of the frequent items and, hence, in the determination of the rules.

The temporal association rules introduced in Ale and Rossi (2000) are an extension of the nontemporal model. The basic idea is to limit the search for frequent sets of items, or *itemsets*, to the lifetime of the itemset's members. On the other hand, to avoid considering frequent an itemset with a very short period of life (for example, an item that is sold once), the concept of temporal support is introduced. Thus, each rule has an associated time frame, corresponding to the lifetime of the items participating in the rule. If the extent of a rule's lifetime exceeds a minimum stipulated by the user, we analyze whether the rule is frequent in that period. This concept allows us to find rules that with the traditional frequency viewpoint, it would not be possible to discover.

The lifespan of an itemset may include a set of subintervals. The subintervals are those such that the given itemset: (a) has maximal temporal support and (b) is frequent. This new model addresses the solution of two problems: (1) Itemsets not frequent in the entire lifespan but just in certain subintervals, and (2) the discovery of every itemset frequent in, at least, subintervals resulting from the intersection of the lifespans of their components, assuring in this way the anti-monotone property (Agrawal & Srikant, 1994). Because of this we call "general" the rules formed from these kinds of frequent itemsets.

## BACKGROUND

Previous work about data mining that includes temporal aspects is usually related to the sequence of events' analysis (Agrawal & Srikant, 1995; Bettini, Wang, Jajodia, & Lin, 1998; Mannila, Toivonen, & Verkamo, 1995). The

usual objective is to discover regularities in the occurrence of certain events and temporal relationships between the different events. In particular, in Mannila et al. the authors discuss the problem of recognizing frequent episodes in an event sequence; an episode is defined there as a collection of events that occur during time intervals of a specific size. Meanwhile Agrawal and Srikant (1995) review the problem of discovering sequential patterns in transactional databases. The solution consists in creating a sequence for every customer and to look for frequent patterns into each sequence. In Bettini et al. the authors consider more complex patterns. In these cases temporal distances with multiple granularities are treated. Chakrabarti, Sarawagi, and Dom (1998), in a totally different approach, use the minimum description length principle together with an encoding scheme to analyze the variation of inter-item correlation along time. That analysis, whose goal is extracting temporally surprising patterns, is an attempt to substitute the role of knowledge domain in searching interesting patterns.

Now we will analyze how the present work is related to others, specifically in mining temporal association rules. All of them have the same goals as ours: the discovery of association rules and their periods or interval time of validity. Our proposal was formulated independently of the others but shares with them some similarities. In Ozden, Ramaswamy, and Silberschatz (1998), the authors study the problem of association rules that exist in certain time intervals and display regular cyclic variations over time. They present algorithms for efficiently discovering what they called “cyclic association rules.” It is assumed that time intervals are specified for the user.

In Ramaswamy, Mahajan, and Silberschatz (1998), the authors study how the association rules vary over time, generalizing the work in Ozden et al. (1998). They introduce the notion of calendar algebra to describe temporal phenomena of interest to the users and present algorithms for discovering “calendric association rules,” that is, association rules that follow the temporal patterns set forth in the user-supplied calendar expressions.

The third study (Chen, Petrounias, & Heathfield, 1998) also suggests calendar time expressions to represent temporal rules. They present only the basic ideas of the algorithms for discovering the temporal rules. The fourth study (Li, Ning, Wang, & Jajodia, 2001) is the most elaborated expression within the calendar approach. The authors define calendar schemas and temporal patterns in these schemas. They also define two types of association rules: precise-match and fuzzy-match. They try to find association rules within the time intervals defined by the schemas.

Finally, in Lee, Lin, and Chen (2001) and Lee, Chen, and Lin (2003), the authors introduce some kind of temporal rules and propose an algorithm to discover temporal

association rules in a publication database. The basic idea is to partition the publication database in light of exhibition periods of items. Possibly, this work is the most similar to ours. The notion of exhibition period is similar to our lifespan (Ale & Rossi, 2000) for an itemset, and the same happens with the concept of the maximal common exhibition period that we have called again lifespan when applying the concept to an itemset. The differences with the present work (Ale & Rossi, 2002) are more significant because we are defining frequent subintervals within an itemset’s lifespan and, in this way, we get the a priori property holds.

Our approach is based on taking into account the items’ period of life, or lifespan, this being the period between the first and the last time the item appears in transactions in the database. We compute the support of an itemset in the interval defined by its lifespan or subintervals contained in it, and define temporal support as the minimum interval width. We consider the history of an itemset as a time series, having the possibility of performing on it different kinds of analysis, based on its wavelet transform. However, this last part is not included in this article because of space restrictions. Our approach differs from the others in that it is not necessary to impose intervals or calendars since the lifespan is intrinsic to the data. Even more, we find association rules and the time when they hold, so the calendar approach becomes a special case. Moreover, our model satisfies the downward closure property, which a-priori-based algorithms are based on.

## THE GENERAL TEMPORAL MODEL

Let  $T = \{ t_0, t_1, t_2, \dots \}$  be a set of times over which a linear order  $<_T$  is defined, where  $t_i <_T t_j$  means  $t_i$  occurs before or is earlier than  $t_j$  (Tansel et al., 1993). We will assume that  $T$  is isomorphic to  $\mathbb{N}$  (natural numbers) and restrict our attention to closed intervals  $[t_i, t_j]$ .

Let  $\mathbf{R} = \{ A_1, \dots, A_p \}$ , where the  $A_i$ ’s are called *items*, a *transaction database*  $\mathbf{d}$  is a collection of subsets of  $\mathbf{R}$ .

Each transaction  $\mathbf{s}$  in  $\mathbf{d}$  is a set of items such that  $\mathbf{s} \subseteq \mathbf{R}$ . Associated to  $\mathbf{s}$  we have a time stamp  $t_s$ , which represents the valid time of transaction  $\mathbf{s}$ .

**Example 1.1.a:**  $\mathbf{R} = \{ A, B, C, D, E, F, G, H, I \}$ .  $\mathbf{d}$  is the collection of 10 transactions with tids 100, ..., 1000 and time stamps 1, ..., 10, respectively; see Figure 1(a).

We consider  $\mathbf{d}$  is temporally ordered. Every item has a period of life, or *lifespan*, in the database, which explicitly represents the temporal duration of the item information, i.e., the time in which the item is relevant to the user. The lifespan of an item  $A_i$  is given by an interval  $[t_1, t_2]$ , with

Figure 1. Example 1—The transaction database and the set of 1-item-set candidates

(a) The database <b>d</b>			(b) The lifespan(LS) and support (Sup) for the Items (or 1-itemsets), and frequent lifespan (FLS): the subintervals and corresponding support. This is the set of candidate itemsets.		
<u>T</u>	<u>Tid</u>	<u>Items</u>	<u>Itemset</u>	<u>LS/Sup</u>	<u>FLS:Subintervals/Support</u>
1	100	ABCFHI	A	[1,10],0.5	<[1,10], 0.5>
2	200	ABCG	B	[1,9], 0.44	<[1,4],0.5>,<[6,9], 0.5>
3	300	CDI	C	[1,10], 0.7	<[1,10], 0.7>
4	400	ACI	D	[3,10], 0.37	<[3,6], 0.5>
5	500	DEHI	E	[5,9], 0.4	-----
6	600	AF	F	[1,6], 0.33	-----
7	700	BCI	G	[2,8], 0.29	-----
8	800	CHG	H	[1,8], 0.37	<[5,8], 0.5>
9	900	BE	I	[1,7], 0.71	<[1,7], 0.71>
10	1000	ACD			

$\tau = 3, \sigma = 0.5$

$t_1 \leq t_2$ , where  $t_1$  is the time stamp of the first transaction in **d** that contains  $A_i$ , and  $t_2$  is the time stamp of the last transaction in **d** that contains  $A_i$ .

With each item  $A_i$  and database **d**, we associate a lifespan defined by a time interval  $[A_i, t_1, A_i, t_2]$  or simply  $[t_1, t_2]$  if  $A_i$  is understood. The lifespan of  $A_i$  is noted as  $l_{A_i}$  and it can be defined as the minimal interval I (w.r.t. set inclusion) satisfying for each time t, if t is associated to a transaction containing A, then t is in I. In addition, we define  $l_d$ , the lifespan of **d**, as  $l_d = \cup l_{A_i}, \forall i$ .

**Example 1.1.b:** Observe, for instance,  $l_A = [1,10], l_D = [3,10], l_F = [1,6]$ , and so on, in Figure 1(b) in the column headed by LS/Sup.

The set of transactions in **d** that contain **X** is indicated by  $V(X) = \{s | s \in d \wedge X \subseteq s\}$  (we omit **d** for the sake of clarity). If the cardinality of **X** is k, **X** is called a k-item-set.

We can estimate the lifespan of a k-item-set **X**, with  $k > 1$ , by  $[t, t']$  where  $t = \max\{t_i | [t_1, t_2]$  is the lifespan of an item  $A_i$  in **X** and  $t' = \min\{t_j | [t_1, t_2]$  is the lifespan of an item  $A_j$  in **X** }.

**Example 1.2.a:** In Figure 2(b) we have the 2-item-set candidates with their computed lifespan, such as  $l_{AB} = [1,9], l_{AD} = [3,6], l_{DI} = [3,7]$ , etc.

Let  $X \subseteq R$  be a set of items and  $l_X$  its lifespan. If **d** is the set of transactions of the database, then  $d_{l_X}$  is the subset of transactions of **d** whose time stamps  $t_i \in l_X$ .

With  $|d_{l_X}|$  we indicate the number of transactions of  $d_{l_X}$ . The inclusion of time allows us to determine if an itemset is frequent by computing the ratio between the number of transactions that contain the itemset and the number of transactions in the database, such that their valid time is included in the itemset's lifespan.

Figure 2. Example 1—The frequent 1-item-sets and the 2-item-set candidates

(a) The frequent 1-itemsets, their lifespans(LS) and their frequent life spans and corresponding support.			(b) The 2-item-set candidates, life spans and support, and frequent life spans and support		
<u>Itemset</u>	<u>LS</u>	<u>FLS:Subintervals/support</u>	<u>Itemset</u>	<u>LS/Sup</u>	<u>FLS:Subintervals/Support</u>
A	[1,10]	<[1,10],0.5>	AB	[1,9], 0.22	<[1,4],0.5>
B	[1,9]	<[1,4],0.5>,<[6,9],0.5>	AC	[1,10], 0.4	<[1,6], 0.5>
C	[1,10]	<[1,10], 0.7>	AD	[3,6], 0.0	
D	[3,10]	<[3,6],0.5>	AH	[5,8], 0.0	
H	[1,8]	<[5,8],0.5>	AI	[1,7], 0.28	<[1,4], 0.5>
I	[1,7]	<[1,7], 0.71>	BC	[1,9], 0.33	<[1,4], 0.5>
			BI	[1,7], 0.28	
			CD	[3,10], 0.25	
			CH	[1,8], 0.25	
			CI	[1,7], 0.57	<[1,7], 0.57>
			DI	[3,7], 0.4	<[3,6], 0.5>
			HI	[1,7], 0.28	

$\tau = 3, \sigma = 0.5$



Given an itemset  $X$ , the *temporal support* of  $X$  is the lifespan's amplitude of  $X$ , namely  $|l_X|$ .

We also define a *threshold* for the temporal support: if  $l_d$  is the lifespan of the database and  $|l_d|$  is its duration, then the threshold of the temporal support  $\tau$  is a fraction of  $|l_d|$ . On the other hand, the user could specify a time instant  $t_0$ , such that any item whose lifespan is  $[t_1, t_2]$  and  $t_2 < t_0$  is considered obsolete.

In certain cases, an itemset may not be frequent in the interval corresponding to its entire period of life but just in one or more subintervals of this period. Then, it could participate in interesting rules in these portions of its lifespan. These subintervals do not depend strictly on the data but on the parameters, i.e., the threshold for the temporal support  $\tau$  and the minimum support  $\sigma$ , provided by the user. We will not be interested in every possible subinterval but only in those that are maximal with respect to the temporal support and are frequent in their time frame. Another way for defining the subintervals is taking the subsets of contiguous time units, according to the granularity selected (for instance, days or weeks), in which the itemset has enough frequency, and whose union is not less than  $\tau$ . We will use the first option.

We use  $\tau = 3$  and  $\sigma = 0.5$  in the example. We could have fixed  $t_0 = 5$ ; thus, no item would be considered obsolete since their lifespans' upper limits are all greater than 5.

Given an itemset  $X$ , the *frequent lifespan* of  $X$  ( $fl_X$ ) is the set of subintervals, contained in its lifespan, in which the itemset is frequent.

We can exemplify this last case with the item B that has  $fl_B = \{[1, 4], [6, 9]\}$ , in Figure 1(b).

As set operations are valid over lifespans, the lifespan  $l_X$  of the  $k$ -item-set  $X$ , where  $X$  is the union of the  $(k-1)$ -itemsets  $V$  and  $W$  with lifespans  $l_V$  and  $l_W$ , respectively, is given by  $l_X = l_V \cap l_W$ . The same is not always true for frequent lifespans because of the restrictions of minimum frequency.

The *support* of  $X$  in  $d$  over its lifespan  $l_X$ , denoted  $s(X, l_X)$  (we again omit  $d$  for the sake of clarity), is the set of fractions of transactions in  $d$  that contain  $X$  in every interval maximal corresponding to  $l_X$ . For each subinterval  $[t, t']$  we compute the support as  $|V(\bar{X}, [t, t'])| / |d_{[t, t']}|$ . Given a threshold of support  $\sigma \in [0, 1]$  and a threshold of

temporal support  $\tau$ ,  $X$  is *frequent* in its lifespan  $l_X$  if there exists at least one subinterval  $[t, t']$  in  $l_X$  such that  $s(X, [t, t']) \geq \sigma$  and  $|[t, t']| \geq \tau$ , and  $[t, t']$  maximal in  $l_X$ . In this case, it is said that  $X$  has *minimum support* in  $l_X$ .

In some cases  $fl_X$  contains a single subinterval, and this may be equal or smaller than the period of life for  $X$ .

The example in Figure 2(a) shows different cases: Itemset A with lifespan  $[1, 10]$  is frequent in its entire period of life and its support is 0.5; itemset B with lifespan  $[1, 9]$  is frequent in two subintervals, that is,  $[1, 4]$  and  $[6, 9]$ , both with support 0.5; itemset D with lifespan  $[3, 10]$  is frequent just in the subinterval  $[3, 6]$ , with support 0.5. In Figure 3(a) we can observe, for instance, 2-item -set AB with lifespan  $[1, 9]$ , which is frequent in  $[1, 4]$  and its support is 0.5; itemset DI with lifespan  $[3, 7]$  and frequent lifespan  $[3, 6]$  and support 0.5.

For the sake of clarity we have included Table 1 with the main notation.

A *general temporal association rule* for  $d$  is an expression of the form  $X \Rightarrow Y : fl_{X \cup Y}$ , where  $X \subseteq R$ ,  $Y \subseteq R \setminus X$ , and  $fl_{X \cup Y}$  is the frequent lifespan of  $X \cup Y$ , in a granularity determined by the user.

**Example 2:** Given the frequent itemset  $ABC$ , we are able to consider different rules such as  $AB \Rightarrow C : \{[1, 4]\}$ ,  $AC \Rightarrow B : \{[1, 4]\}$ , etc. Also, we could consider  $I \Rightarrow C : \{[1, 7]\}$  and  $C \Rightarrow I : \{[1, 7]\}$ .

The *confidence of a rule*  $X \Rightarrow Y : fl_{X \cup Y}$ , denoted by  $conf(X \Rightarrow Y : fl_{X \cup Y})$ , is the conditional probability that a transaction of  $d$ , randomly selected in the frequent lifespan  $fl_{X \cup Y}$ , that contains  $X$  also contains  $Y$ .

The conditional probability varies according to the subinterval considered within  $fl_{X \cup Y}$ . Then it is expressed as:

$$conf(X \Rightarrow Y : fl_{X \cup Y}) = \{ s(X \cup Y, [t, t']) / s(X, ([t, t']) | [t, t'] \subseteq fl_{X \cup Y} \}$$

where

$$fl_{X \cup Y} = \{ [t_1, t_2] / |[t_1, t_2]| \geq \tau \text{ and } s(X \cup Y, [t_1, t_2]) \geq \sigma \text{ and } \neg \exists [t_j, t_k] ( [t_1, t_2] \subset [t_j, t_k] \text{ and } s(X \cup Y, [t_j, t_k]) \geq \sigma ) \}$$

**Example 3:** We compute the confidence of the rule  $C \Rightarrow I : \{[1, 7]\}$  as:

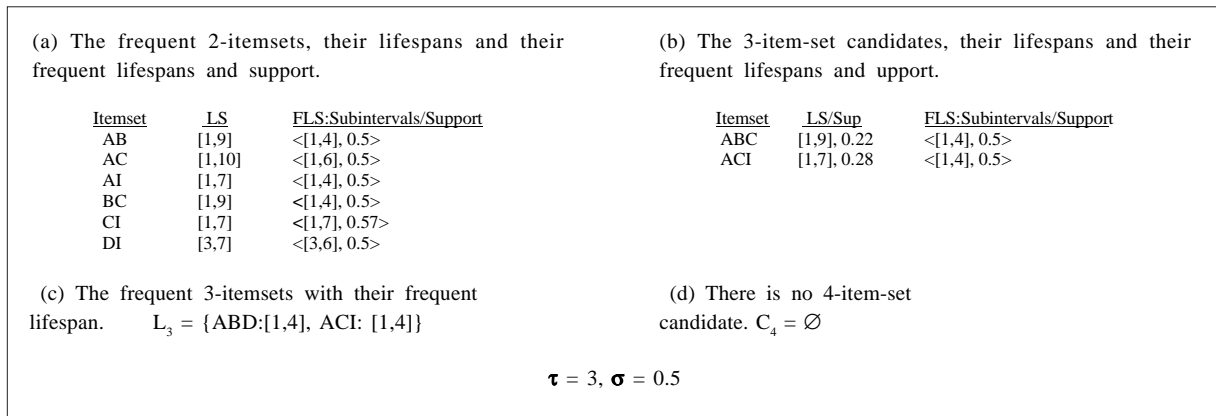
$$Conf(C \Rightarrow I : \{[1, 7]\}) = s(CI, [1, 7]) / s(C, [1, 7]) = 0.57 / 0.71 = 0.80.$$

The general temporal association rule  $X \Rightarrow Y : fl_{X \cup Y}$  holds in  $d$  with support  $s_1, \dots, s_p$ , temporal support  $|fl_{X \cup Y}|$  and confidence  $c_1, \dots, c_p$ , if  $s_1\%, \dots, s_p\%$  of the transactions

Table 1. Notation

$l_X$	life span of the item set $X$
$d _X$	subset of transactions in $d$ with time stamps in $l_X$
$\tau$	threshold for temporal support or minimum temporal support
$\sigma$	threshold for support or minimum support
$\theta$	threshold for confidence or minimum confidence
$t_0$	threshold for obsolescence
$fl_X$	frequent life span for item set $X$
$s(X, l_X)$	support of $X$ in its life span $l_X$

Figure 3. Example 1—The frequent 2-item sets, the 3-item sets candidates and the frequent 3-item sets



of  $\mathbf{d}$  in  $fl_{X \cup Y} = \{ \text{subinterval}_1, \dots, \text{subinterval}_p \}$  contain  $\mathbf{X} \cup \mathbf{Y}$  and  $c\%_1, \dots, c\%_p$  of the transactions of  $\mathbf{d}$  that contain  $\mathbf{X}$  also contain  $\mathbf{Y}$  in the set of time frames  $[t, t']$  such that  $[t, t'] \subseteq fl_{X \cup Y}$ .

**Example 4:** The rule  $C \Rightarrow I : \{[1,7]\}$  holds in  $\mathbf{d}$  with support 0.57, temporal support 7, and confidence 0.80.

The discovery of all the association rules in a transaction set  $\mathbf{d}$  can be made in two phases (Agrawal, Imielinski, & Swami, 1993). In the following paragraph, we introduce suitable modifications to support general temporal association rules discovery.

**Phase 1T:** Find every itemset  $X \subseteq R$  such that  $X$  is frequent in its lifespan  $l_x$ , i.e.,  $s(X, [t, t']) \geq \sigma, |[t, t']| \geq \tau$  and  $[t, t']$  maximal, for  $[t, t']$  in  $l_x$ , that is,  $fl_x \neq \emptyset$ .

**Phase 2T:** Use the frequent itemsets  $X$  to find the rules: Verify for every  $Y \subset X$ , with  $Y \neq \emptyset$  if the rule  $X \setminus Y \Rightarrow Y : fl_{X \cup Y}$  is satisfied with enough confidence, in other words, exceeds the minimum confidence  $\theta$  established in the interval  $[t, t]$  for all  $[t, t']$  in  $fl_{X \cup Y}$ .

## FUTURE TRENDS

Traditionally temporal data mining is applied to static and regular temporal data sets. But real-world data might be complex and produced in huge amounts in real time. These new aspects of temporal data will deserve new theories and algorithms which should cope with the following.

- **Data streams:** Some temporal data is stored only temporally and requires real-time analysis; in par-

ticular, association rules analysis over data just produced by communication switches, POS devices, etc.

- **Heterogeneous data types:** Temporal data is usually expressed as partly categorical events and partly numerical time series. There exists a need to analyze all possible data in a uniform way.

On the other side, in order to improve our capabilities of analysis, it is necessary to enrich the algorithms with knowledge from other sources such as Markov-process-based modeling and other modeling techniques. It will benefit areas such as applications in bioinformatics: We believe that high-performance temporal data mining tools will play a crucial role in the analysis of the ever-growing databases of biosequences/biostructures.

## CONCLUSION AND FUTURE WORK

We have presented a model for the discovery of general temporal association rules. Each itemset has an associated lifespan, which comes from the explicitly defined time in database transactions. In particular, each frequent itemset has an associated frequent lifespan, that is, a set of intervals in which it is frequent.

As immediate future work, we will analyze the problem of the maintenance of temporal association rules.

In another direction, we are studying the problem of *trend dependencies*. We analyze a historical database and try to discover all the trend dependencies. For this, we find all the frequent itemsets in their lifespan and then analyze their temporal trends.

We are also interested in finding substitute items in *market basket analysis*. Under certain conditions, it is

possible to discover pair of rules which differ in just one item and are complementary with respect to the lifespan of these different items. In this case, we say that such items are substitute one for each other.

## REFERENCES

Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. *Proceedings of ACM SIGMOD* (Vol. 22, pp. 207-216).

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. *IBM Res. Rep. RJ9839*, IBM Almaden.

Agrawal, R., & Srikant, R. (1995). Mining sequential patterns. *Proceedings of 11<sup>th</sup> IEEE International Conference on Data Engineering* (pp. 3-14).

Ale, J., & Rossi, G. (2000). An approach to discovering temporal association rules. *Proceedings of ACM 15<sup>th</sup> Symposium on Applied Computing*, (Vol. 1, pp. 294-300).

Ale, J., & Rossi, G. (2002). The itemset's lifespan approach to discovering general temporal association rules. *Proceedings ACM 2nd Temporal Data Mining Workshop* (pp. 1-10).

Bettini, C., Wang, X., Jajodia, S., & Lin, J. (1998). Discovering frequent event patterns with multiple granularities in time sequences. *IEEE TOKDE*, 10(2), 222-237.

Chakrabarti, S., Sarawagi, S., & Dom, B. (1998). Mining surprising patterns using temporal description length. *Proceedings 24<sup>th</sup> VLDB Conference*.

Chen, X., Petrounias, I., & Heathfield, H. (1998). Discovering temporal association rules in temporal databases. *Proceedings International Workshop IADT'98*.

Lee, C. H., Chen, M. S., & Lin, C. R. (2003). Progressive partition miner: An efficient algorithm for mining general temporal association rules. *IEEE Transactions on Knowledge and Data Engineering*, 15(4), 1004-1017.

Lee, C. H., Lin, C. R., & Chen, M. S. (2001). On mining general temporal association rules in a publication database. *Proceedings of the IEEE International Conference on Data Mining (ICDM-01)*.

Li, Y., Ning, P., Wang, X., & Jajodia, S. (2001). Discovering calendar-based temporal association rules. *Proceedings 8th International Symposium on Temporal Representation and Reasoning* (pp. 111-118).

Mannila, H., Toivonen, H., & Verkamo, I. (1995). Discovering frequent episodes in sequences. *KDD'95. AAAI* (pp. 210-215).

Ozden, B., Ramaswamy, S., & Silberschatz, A. (1998). Cyclic association rules. *ICDE 1998*.

Ramaswamy, S., Mahajan, S., & Silberschatz, A. (1998). On the discovery of interesting patterns in association rules. *Proceedings 24th VLDB Conference*.

Tansel, A. U., Clifford, J., Gadia, S. K., Jajodia, S., Segev, A., & Snodgrass, R. T. (Eds.). (1993). *Temporal databases: Theory, design, and implementation*. Benjamin/Cummings.

## KEY TERMS

**Association Rule:** A statement  $A \Rightarrow B$ , which states that if A is true then we can expect B to be true with a certain degree of confidence. A and B are sets of items, and the  $\Rightarrow$  operator is interpreted as "implies."

**Association Rule Mining:** The data mining task of finding all association rules existing in a database, having support and confidence greater than a minimum support value and a minimum confidence value.

**Confidence of a Rule:** Percentage of the rows that contain the antecedent that also contain the consequent of the rule. The confidence of a rule gives us an idea of the strength of the influence that the antecedent has on the presence of the consequent of the rule.

**Data Mining:** The nontrivial extraction of implicit, previously unknown, and potentially useful information from data.

**Lifespan:** The time over which a database object is defined.

**Lift:** A measure used to determine the value of an association rule that tells us how much the presence of the antecedent influences the appearance of the consequent.

**Market Basket Analysis:** The process of looking at the transaction or market basket data to determine product affinities for each item in the basket.

**Support of a Rule:** The fraction of the rows of the database that contain both the antecedent and the consequent of the rule. The support of a rule tells us in how many instances (rows) the rule can be observed.

**Temporal Database:** A database that supports some aspect of time, not counting user-defined time.

# Document Versioning in Digital Libraries

**Mercedes Martínez-González**  
*Universidad de Valladolid, Spain*

## INTRODUCTION

Digital libraries are systems that contain organized collections of objects, serving in their most basic functions as a mirror of the traditional library that contains paper documents. Most of the information contained in the collections of a digital library consists of documents, which can evolve with time. That is, a document can be modified to obtain a new document, and digital library users may want access to any of those versions. This introduces in digital libraries the problem of versioning, a problem that has also been considered in a related community, the hypertext community (hypermedia in its most extensive acceptance). Some domains in which document evolution is very important are the legislative domain (Arnold-Moore, 2000; Martínez-González, 2001; Vitali, 1999), the management of errata made to scientific articles (Poworotznek, 2003) and software construction (Conradi & Westfechtel, 1998).

In the legislative domain, rules suffer amendments that result in new versions of the amended rules. Access to all versions of a document is an important facility for their users; for example, to understand a tribunal sentence it is necessary to get access to the text of involved rules, as they were valid at the moment the sentence was made. In legislative documents, modifications are embedded inside other documents, so that the document to be modified is cited and how it should be modified is expressed later. For each modification, its author cites the document fragment he or she wants to change and indicates how the said fragment could be modified (e.g., eliminating it, substituting it). The new version obtained by the application of these changes is virtual, in the sense that the library users<sup>1</sup> know it exists but there is no physical copy of it available.

Figure 1 shows an example. The text that appears in the figure is a fragment of an EU normative document. It is possible to recognise here the 20th article of the document, that modifies another article of a different document (the '1968 convention'). The reference in the figure indicates in a precise manner the article (number 41) affected by the modification (substitution) that follows. The legislators leave to the readers the cut-and-paste work needed to obtain an updated version of the modified convention.

Errata to scientific articles are somehow similar. The errata are posterior to the original article and they are published together with the reference to the part of the article to be changed by the modification. How and where the errata are inserted varies among publishers and among publications from the same publisher. One way to insert errata is by listing it at the beginning or at the end of the corrected article.

Software construction is a bit different, as it is the composition of software with several of the program files considered here. The different versions of program files are available at the same time, and the composition of software has to assemble adequate versions in order to obtain the correct version of the software. There is no need to be precise about the internal parts of a document or file affected by changes; the program files used to obtain a software are not fragmented during the composition process.

Next, the issues related with document versioning are revised, the main approaches are proposed, and the issues that each approach privileges are identified. Some of them are more recent than others, but promising. Versioning a document impacts not only the document itself but also other items, as references from and to the versioned document, or the indexes created for information retrieval operation.

*Figure 1. A fragment of a modifier EU normative document<sup>2</sup>*

Article 20  
 The following shall be substituted for Article 41 of the 1968 Convention:  
 "Article 41  
 A judgement given on an appeal provided for in Article 40 may be contested only:  
 - in Belgium, France, Italy, Luxembourg and the Netherlands, by an appeal in cassation,  
 - in Denmark, by an appeal to the *højesteret*, with the leave of the Minister of Justice,  
 - in the Federal Republic of Germany, by a *Rechtsbeschwerde*,  
 - in Ireland, by an appeal on a point of law to the Supreme Court,  
 - in the United Kingdom, by a single further appeal on a point of law."

## BACKGROUND

As for the issues of interest related to document versions, there are seven categories:

1. **What can be versioned:** This question can be considered from two perspectives. The first perspective considers objects stored in the system. This is the typical situation in the Web and hypertext environments. Hypertext nodes (e.g., documents, files) can change, but the hypertext structure can also change (e.g., objects may vary their location, some of them may disappear, others may change their references to other objects). The evolution considered in the second perspective is the one used by digital library users—these documents may or may not match unidirectionally any of the objects stored in the digital library (Arms, Blanchi, & Overly, 1997). Changes in this case affect the content of documents: the text, the internal structure of documents (e.g., this is the case for structured documents), or references (i.e., citations within documents that are part of document content). How the evolution of documents at the user level impacts the system depends on the technical solutions used in the digital library (a digital library can use hypertext models and software, textual document databases, or other types of databases).
2. **Detecting changes:** Sometimes it is necessary to recognise two documents as versions of the same work or to find the changes that have to be applied to a document version to obtain another one. There are two possible ways to do this: extracting references from document content (Martínez-González, de la Fuente, Derniame, & Pedrero, 2003; Thistlewaite, 1997), or comparing versions (Chawathe, Rajaraman, García-Molina, & Widom, 1996; Cobena, Abiteboul & Marian, 2002; Lim & Ng, 2001).
3. **Representing changes:** The information about versions and their differences has to be stored somehow in the system. There are three ways to accomplish it:
  - To store the versions caused by a change or the corresponding differences. This is the solution 1 of versioning management approaches.
  - To represent changes as annotations (attributes) to the affected nodes. This is solution 2 for version management. All the history of a node is described in these annotations.
  - To model modification relationships as links between the modifiers and the target of the modification (Martínez-González et al., 2003). This solution considers the semantic relationship that is

behind a change.

4. **Querying changes or querying the history of a document:** This consists of answering questions about a document evolution, such as “What are the documents that modify this one?” or “What are the modifications of this document over the last month?” The way to operate here is dependent on the choice made for representing changes. If versions are first class objects, the only way to query changes is to search for differences between versions. It can be a file or tree (in case of structured documents) comparison. If changes are represented as annotations, to query them is to query annotations. In the case of structured documents, that means to query node attributes. If changes are represented as independent links, to query changes is to query links, which is the same as querying any document.
5. **Accessing any version:** The access to any version can be provided either by storing all of them or by composing them on demand. It depends on the approach chosen for version management.
6. **Dealing with the propagation (effects) of versioning some items on related items:** For example, changes to a document affect references or links that reach the document. This is the well-known problem of dangling links common in the Web or, in a more general definition, the referential integrity issue (Ashman, 2000). Other possible impacts are on indexes used for information retrieval operations.
7. **Inferring the composition rules of new versions:** In certain situations, such as the example in Figure 1, the new versions are virtual (no copy or direct way to obtain it is provided), and the structure of the new version has to be inferred from the information provided about modifications. Humans commonly assume this task, but an automatic inference can also be tackled (Martínez-González, de la Fuente, & Derniame, 2003).

## APPROACHES TO THE MANAGEMENT OF VERSIONS

Different approaches can be distinguished. Here they are listed, ordered temporally: The ones with longer tradition appear first and the more recent ones are at the end of the list.

1. **Maintaining simultaneously all versions (or deltas) in digital library collections and linking related versions:** The main problem with this ap-



proach is links maintenance (Choquette, Poulin, & Bratley, 1995). This approach facilitates access to any version and does not consider queries about the evolution of versioned items. The propagation of versioning effects is considered but there are no general solutions, and this is still a difficult issue to deal with. This approach and variations of it have received a good amount of attention in the version control area.

2. **Considering different stamps of the database and comparing them to detect changes reflecting that an object has been versioned (Cellary & Jomier, 1992):** This solution is used with object databases and therefore can be considered when modelling documents as objects (Abiteboul et al., 1997). In this approach, changes are represented indirectly as the difference between two database states.
3. **Modelling modifications as attributes and storing this information with documents:** This approach comes from the area of semistructured data, which include structured documents. Changes are represented as annotations (attributes) to the affected nodes, facilitating queries about nodes "history." The detection of versions is done by tree comparisons (Chawathe et al., 1996; Cobena et al., 2002). In contrast to the previous solutions, this is the first in which document structure is considered, thereby associating changes to document fragments instead of to whole documents. It is possible to know that the document has been changed and where to find the changes; however, it is up to the user to obtain the versions if this is his or her wish. For the same reason, it does not facilitate the automatic composition of versions.
4. **Automatically composing versions of documents:** This can be done by keeping the rules that allow the generation of versions of documents. This is the option named *intentional versioning* (Conradi & Westfechtel, 1998) and has been used for document management (Arnold-Moore, 1997). It is also possible to compose versions by querying metadata (i.e., attributes, links) stored in the system databases (Hemrich, 2002; Martínez-González et al., 2003). These solutions deal well with access to versions and they are in a good position to treat queries about version evolution. Their main weakness is dealing with the propagation of versioning effects on information retrieval operations.

## MANNERS OF IMPLEMENTING VERSIONING SOLUTIONS

The manners to implement solutions to manipulate versions have evolved with time. First, there are the version

control servers, which provide general mechanisms for version control issues (Hicks, Legget, Nürnberg, & Schnase, 1998; Whitehead, 2001). As for the manner to represent and store the versioning information, some solutions using HTML elements (Vitali, 1999) were proposed. This is a possible implementation of modelling changes in attributes, which has been superseded with the arrival of XML. A more recent option consists of automatically composing versions. This type of solution appeared with the arrival of XML. There are variations of doing this, such as splitting documents into pieces and storing the composition rules of each version (Arnold-Moore, 2000). Instead of storing composition rules, they can be obtained by querying attributes describing the temporal validity of pieces of content (Hemrich, 2002). Another option is storing the information about modification relationships between documents and inferring the structure (composition rules) and content of a new version from the information stored in the modifications database (Martínez-González, 2001).

## FUTURE TRENDS

Some issues related with document versioning, such as querying document changes or dealing with the propagation of versioning, receive the attention of the scientific community recently as compared with some other aspects, as the access to any version. Thereafter, these are open questions, as the solutions proposed until now are not yet satisfactory or scalable enough (see Ashman, 2000; Cobena et al., 2002 for details about some solutions). With the massive use of the Web, where these problems also appear, it is to expect new proposals to emerge. Of course, digital libraries will benefit from this, as they would improve the quality of the services offered to their users.

## CONCLUSION

Document evolution demands digital libraries to provide solutions to manipulate versions and to satisfy user requests. Several issues emerge related to versioning, including accessing any version of a document, querying its history, managing the impact of versioning an item on related items, and so forth. The background of dealing with these problems is varied. The version control community has long studied issues as access to any version. They also know well the problems that versioning hypertext elements may cause on other hypertext items as links. However, this study area does not consider other issues, such as querying the history of a document

or the impact of versioning on information retrieval operations as indexing.

More recent are the approaches that compose versions automatically and infer the composition rules (structure) of versions from semantic information. These solutions, which introduce dynamism and knowledge extraction in version management applications, are promising for digital libraries. They can help to solve automatically some issues that otherwise could not be treated in many of these systems, because a manual introduction of composition rules or versioning information is, in many cases, unaffordable.

## REFERENCES

- Abiteboul, S., Cluet, S., Christophides, V., Milo, T., Moerkotte, G., & Simeon, J. (1997). Querying documents in object databases. *International Journal on Digital Libraries*, 1(1), 5-19.
- Arms, W. Y., Blanchi, C., & Overly, E. A. (1997, February). An architecture for information in digital libraries. *D-Lib Magazine*.
- Arnold-Moore, T. (1997). Automatic generation of amendment legislation. *Sixth International Conference on Artificial Intelligence and Law, ICAIL'97*, Melbourne, Victoria, Australia.
- Arnold-Moore, T. (2000). Connected to the law: Tasmanian legislation using EnAct. *Journal of Information, Law and Technology*, 1. Retrieved September 7, 2004, from <http://elj.warwick.ac.uk/jilt/01-1/>
- Ashman, H. (2000). Electronic document addressing: Dealing with change. *ACM Computing Surveys*, 32(3), 201-212.
- Cellary, W., & Jomier, G. (1992). Consistency of versions in object-oriented databases. In F. Bancilhon et al. (Eds.), *Building an object-oriented database system. The story of O<sub>2</sub>: Vol. 19. The Morgan Kaufmann Series in Data Management Systems* (pp. 447-462). Morgan Kaufmann.
- Chawathe, S., Rajaraman, A., Garcia-Molina, H., & Widom, J. (1996). Change detection in hierarchically structured information. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 25(2), 493-504.
- Choquette, M., Poulin, D., & Bratley, P. (1995). Compiling legal hypertexts. In N. Revell & A. M. Tjoa (Eds.), *Database and expert systems applications, 6<sup>th</sup> International Conference, DEXA'95, Lecture Notes in Computer Science*, 978 (pp. 449-58).
- Cobena, G., Abiteboul, S., & Marian, A. (2002). Detecting changes in XML documents. *Data Engineering 2002 (ICDE2002)*, 41-52.
- Conradi, R., & Westfechtel, B. (1998). Version models for software configuration management. *ACM Computing Surveys (CSUR)*, 30(2), 232-282.
- Hemrich, M. (2002). A new face for each show: Make up your content by effective variants engineering. *XML Europe 2002*. Retrieved September 7, 2004, from <http://www.idealliance.org/papers/xml02/>
- Hicks, D. L., Leggett, J. J., Nürnberg, P. J., & Schnase, J. L. (1998). A hypermedia version control framework. *ACM Transactions on Information Systems*, 16(2), 127-160.
- Lim, S. J., & Ng, Y. K. (2001). An automated change-detection algorithm for HTML documents based on semantic hierarchies. *The 17<sup>th</sup> International Conference on Data Engineering (ICDE 2001)*, Heidelberg, Germany.
- Martínez-González, M. (2001). *Dynamic exploitation of relationships between documents in digital libraries: Application to legal documents*. Doctoral dissertation, Universidad de Valladolid, España; Institut National Polytechnique de Lorraine, France.
- Martínez-González, M., de la Fuente, P., & Derniame, J.-C. (2003). XML as a means to support information extraction from legal documents. *International Journal of Computer Systems Science and Engineering*, 18(5), 263-277.
- Martínez-González, M., de la Fuente, P., Derniame, J., & Pedrero, A. (2003). Relationship-based dynamic versioning of evolving legal documents. *Web-knowledge Management and Decision Support, Lecture Notes on Artificial Intelligence*, 2543, 298-314.
- Poworotznek, E. (2003). Linking of errata: Current practices in online physical sciences journals. *Journal of the American Society for Information Science and Technology*, 54(12), 1153-1159.
- Thistlewaite, P. (1997). Automatic construction and management of large open webs. *Information Processing and Management*, 33(2), 161-173.
- Vitali, F. (1999). Versioning hypermedia. *ACM Computing Surveys*, 31(4), 24.
- Whitehead, E. J. (2001). Design spaces for link and structure versioning. *Proceedings of Hypertext'01, 12<sup>th</sup> ACM Conference on Hypertext and Hypermedia*, Aarhus, Denmark.

## KEY TERMS

**Digital Library:** A set of electronic documents organized in collections, plus the system that provides access to them. They are the digital version of traditional libraries.

**Hypertext:** The organization of information units as a network of associations, which a user can choose to resolve. Hypertext links are the instances of such associations.

**Intensional Versioning:** Automatic construction of versions based on configuration rules.

**Referential Integrity:** In hypertext, a measure of the reliability of a reference to its endpoints. A reference has the property of referential integrity if it is always possible to resolve it. When references are represented as links it is called *link integrity*.

**Structured Documents:** Documents made by composing well-delimited pieces of content that can present an inclusion hierarchy between them.

**Version Control:** Set of mechanisms that support object evolution in computer applications.

**Versions:** Variations of an object with a high degree of similarity. Document versions are never completely equal, but they are similar enough so as to be recognisable as the same document.

**Virtual Document:** A document (intellectual entity) that exists in the conscience of individuals but of which there is no physical copy available.

**XML:** Extensible markup language. Markup language for structured documents. Structure is represented with textual markup that intermixes with document content. XML is a recommendation from the World Wide Web Consortium (W3C).

## ENDNOTES

- <sup>1</sup> Library users are people who access the digital library searching for documents (intellectual entities) that match their requirements. These users may be specialists in a domain (e.g., jurists), with no special knowledge about computers, who just use the library as another tool for their work.
- <sup>2</sup> Adapted from Convention of Accession of 9 October 1978 of the Kingdom of Denmark, of Ireland and of the United Kingdom of Great Britain and Northern Ireland to the Convention on jurisdiction and enforcement of judgments in civil and commercial matters and to the Protocol on its interpretation by the Court of Justice.

# E-Government Databases

**Catherine Horiuchi**  
*Seattle University, USA*

## INTRODUCTION

The new face of government is electronic. Prior to the development of e-government, adoption of small-scale computing and networking in homes and businesses created the world of e-business, where computer technologies mediate transactions once performed face-to-face. Through the use of computers large and small, companies reduce costs, standardize performance, extend hours of service, and increase the range of products available to consumers. These same technological advances create opportunities for governments to improve their capacity to meet growing public service mandates. Tasks that formerly required a trip to city hall can be accomplished remotely. Government employees can post answers to frequently asked questions online, and citizens can submit complex questions through the same electronic mail (e-mail) systems already used at home and in businesses. This developing e-government increases the number and complexity of electronic databases that must be managed according to the roles information plays in government operations.

## BACKGROUND

E-government has been defined as the application of e-business technologies and strategies to government organizations, the delivery of local government service through electronic means, and a method to enhance the access to and delivery of its services to benefit citizens. E-government is envisioned to be “a tool that facilitates creation of public value” (United Nations, 2003). In its survey of member states, the UN found 173 of 191 members have government Web sites, offering at minimum government information and some measure of service. Common capabilities of governmental Web sites include ability to download government forms and access information on a region or local services (Swartz, 2004). The United States has the largest number of citizens with Internet-connected computers at home and the most developed network infrastructure, resulting in the greatest amount of information and number of services and products available online. Even so, information technology process adaptations for government services are

rudimentary, and many citizens do not have the ability to access their government remotely.

Less visibly, e-government adoption is steadily increasing connections made through the Internet between public agencies and private networks. This connectivity creates computer-to-computer interfaces between multiple databases. Some connections include Web-enabled access to decades-old data structures stored on traditional mainframe systems. Other systems use new databases built or bought from proprietary third parties. A third type of major database project ports data from older systems into new Web-enabled models. Functional requirements determine the database management properties needed by each data system.

## E-GOVERNMENT DATA SYSTEMS

Technology utilization and diffusion of technological processes are increasingly important in managing the “hollow state” (Milward & Provan, 2000). This model of governance is desirable because it allows for greater flexibility in government budgeting and operations. A systematic contracting process creates price competition and encourages fresh approaches to social services, but also requires coordination of information and applications to exchange data securely between the service providers and the government agency. To facilitate this, e-government has adopted structuring concepts first developed in commercial sectors as business-to-business (B2B) and business-to-consumer (B2C) applications. Process re-engineering in government is transforming these concepts into government-to-business, government-to-citizen, and government-to-government constructs with parallel acronyms G2B, G2C, and G2G, respectively. Each of these models creates its own service and reliability profile. Diffusion of data is an essential element of these business processes adopted by government (Bajaj & Sudha, 2003), resulting in large data sets shared among technology project partners.

Sometimes, these diffusions go awry because government is not like business in fundamental ways (Allison, 1980) that make government databases more comprehensive and at the same time more vulnerable to data dissemination contrary to established procedures. In one high



profile case, over five million traveler records were provided by an airline to a government contractor working on a Pentagon data mining and profiling project (Carey & Power, 2003), contrary to the airline’s privacy policy. The data were cross-referenced to other demographic data available from another firm. The expanded data set was then used in a paper that was presented at a technology conference and posted briefly on the conference Web site. In a similar case of unauthorized data exchange, an airline transferred 1.2 million records—names, addresses, phone numbers, and credit-card information—to multiple private firms competing for government contracts. This was not reported for nearly two years (McCartney & Schatz, 2004). The U.S. government’s Computer-Assisted Passenger Prescreening System (CAPPS II), intended solely as a screening tool, was cancelled in response to the public outcry over multiple incidents of unauthorized release of data (Alonso-Zaldivar, 2004). The misuse occurred despite assurances by the government that travel data would be protected from precisely this type of diffusion and dissemination, that “information will only be kept for a short period after completion of the travel itinerary, and then it will be permanently destroyed” (Department of Homeland Security, 2003a, p. 1) and stored “at a TSA secure facility” (Department of Homeland Security, 2003b, p. 16).

Models developed for commercial enterprises weakly map to specific requirements of governmental systems. The myriad roles of government result in unique system requirements. Table 1 lists some of the e-government applications already in operation in at least some jurisdictions.

The range in requirements can be highlighted by a brief comparison of two very different applications and properties of the databases they require. In the adoption of e-voting, governments are replacing paper ballots with touch-screen terminals as the voter interface, using machine technology for tabulating the votes and aggregating election results. Once an election is certified, the database can be reinitialized for the next election. For parcel and tax records, information that has been kept for decades or even centuries on paper is now stored on computers that are subject to periodic replacement cycles. Humankind’s earliest writings and data collections are

comprised of this very same information, so persistence is a fundamental attribute.

The public manager must develop information strategy and management skills better suited to the adoption of advanced technologies (Dawes, 2004). These new skills involve system security models, measuring system efficacy, utilizing tools for archival of public records, and managing information across system life cycles that vary widely, put into a public service context. Public managers have not developed these skills at the same rate that they have implemented new technologies or undertaken large projects, contributing to the high failure rate of public sector technology initiatives. These failures range from high-profile project abandonment to an assessment of low value for cost, such as the 1994 assessment that \$200 billion in federal expenditures over 12 years created little discernable benefit (Brown & Brudney, 1998).

## ELECTRONIC VOTING

Beginning in the 1990s with the development of easily understood Internet browser applications and the widespread penetration of personal computers into households, governments have sought to facilitate access and services through this familiar interface. The introduction of modern technology to one fundamental government activity – voting – has drawn special attention both for its potential to accurately and rapidly count votes and for the specific problems associated with the data collected.

Voting machine technology has gradually been modernized. The original paper ballots have been replaced in varying districts. First developed were mechanical lever machines, then punch cards either supplied with hole punchers or pre-perforated. Optical scanners read cards with ovals or squares that have been filled in by pencil or pen. The most recent introduction incorporates personal computer technology and networks to create computer-based electronic voting (e-voting) systems. All but the latest technologies have been in active use for decades. In the 1996 U.S. presidential election, approximately 2% of the votes were cast on paper ballots, 21% on mechanical lever systems, 37% on punched cards, 25% on optical cards, and nearly 8% on e-voting machines (Hunter, 2001). None of these counts votes perfectly; in 1984, the state of Ohio invalidated 137,000 punch card ballots (Whitman, 2000). Despite ongoing efforts to modernize voting technologies to simplify voting and reduce errors, an election dispute in 2000 began a public debate on computers, databases, networks, and voting.

The contested U.S. presidential election involved the unlikely scenario where the margin of votes for victory in several states appeared smaller than the margin of error for the semi-automated tools that tabulated votes. This re-

Table 1. E-government public services applications

- |  |
|--|
| <ul style="list-style-type: none"> <li>• Informational Web site (online brochure)</li> <li>• Permitting</li> <li>• Webcasting of meetings</li> <li>• License renewals</li> <li>• Records requests</li> <li>• E-voting</li> </ul> |
|--|



sulted in the infamous pictures of election officials from the state of Florida holding up punch cards rejected by their tabulating machine to determine whether or not a voter had at least partially registered a vote. The popular vocabulary expanded to include the technical terms “hanging chad” for a partially detached square and “dimpled chad” where the card was merely dented. Voting officials involved in the recount argued on the “voter intent” for each type of incomplete perforation.

The recount process ultimately settled by the U.S. Supreme Court resulted in heightened interest in replacing older technology, punch cards in particular (Niman, 2004). With electronic touch screen voting, there would be no paper ballots that would be processed through a tabulating machine. Instead, the new machine would count the electronic votes and then send the totals electronically through a computer link. Despite initial claims by vendors that their systems would have higher reliability, serious problems have developed. For example, “Diebold’s voting system... inexplicably gave thousands of Democratic votes in the Oct. 7 recall election to a Southern California socialist” (Hoffman, 2004). While Diebold and other companies alter their machinery to comply with new mandates for paper voter receipts, questions remain on the security and reliability of the computer systems and the database of votes.

## TAX AND PARCEL RECORDS

Governments have long had responsibility for essential data related to ownership of property and collection of taxes. The automation of tax rolls and parcel ownership records creates databases with special requirements for persistence and public disclosure. Under government rules, parcel data and aggregate tax data are generally available for public inspection; advances in software interfaces could result in meeting this public inspection requirement through the Internet. However, parcel records are often linked in legacy mainframe applications to individual tax records that are not available for public inspection, so most local governments do not allow online access to the parcel records, lacking a sure, simple, and affordable method to segment the data structures.

Local governments have undergone gradual adoptions of supplemental databases to manage land use. Many planning departments have automated simple permitting. Geographic information systems (GIS) are implemented across multiple local government boundaries (Haque, 2001). Shared systems in metropolitan zones allow for better cost management of new technologies and create regional models of growth, but these new systems do not have the persistence of the paper maps they replace. An original book of parcel maps may last well over a hundred

years, while at best the life cycle of a computer database is measured in decades. Many hardware and software systems are replaced or migrated to newer technologies every few years; so there is a tendency for governments to replicate data in new systems, rather than create a single multi-function database.

## FUTURE TRENDS

Governments face challenges in their adoption of modern technology systems and supporting databases (Thomas & Streib, 2003). Some factors encourage persistent use of outdated systems in government operations. These include the long duration of governments, civil service protection for government workers, and non-competitive provision of fundamental public safety and other services. Other factors support developing new systems: elections often result in changes in political leadership, altering short-term interests and priorities, while efficiency initiatives for public/private partnerships require new linkage and partitioning of governmental data structures. Adding to these functional challenges are security and privacy constraints (Bond & Whiteley, 1998). Table 2 summarizes security-related threats to electronic databases that public managers must address.

Security concerns have increased as databases grow and networks result in data proliferation. These concerns extend to the capacity of government to ensure privacy of information (Strickland, 2003). Uncertain budgets constrain initial implementation and threaten the maintenance of a system under affordability mandates. Security of systems increases the cost and provides no service-related benefit, and many observers question the efficacy of security and privacy efforts by government agencies.

Table 2. Security challenges of electronic public databases

- Data can be “stolen” through copying, while leaving the source data intact.
- Databases can be encrypted, but this function is often enacted in the absence of unified management of encryption keys.
- Custom systems can result in platform and database software version dependency, limiting application of security patches.
- Back-ups and test restorations limit potential corruption or loss from system failures but proliferate copies of sensitive databases.
- Linking data from legacy systems with newer applications that offer more information may limit inherent referential integrity.

## CONCLUSION

While the mechanics of information management are similar in the public and private sectors, constraints and special requirements in public service modify the role of hardware and software systems, particularly the databases which increasingly contain personal information. These differences require information technology savvy and management skills redefined for the governmental context.

Governments use data structures to manage information on individuals, properties, boundaries, as well as to perform basic operational functions such as accounting. Databases that store government information have widely divergent operating requirements. In e-voting, validity is essential, but the data does not need to persist once the voting results have been certified. Tax and parcel records must persist for decades, if not centuries. Security and privacy are matters of interest to government agencies, but adequate funds may not be available to address concerns which develop as a corollary to database development and network connectivity; limited access and slow development of Web-enabled government are among security and privacy strategies in use. Addressing technical constraints on information technology, meeting security challenges, and satisfying citizen expectations will be keys to the success of the next generation of public managers.

## REFERENCES

- Allison, G. (1980). In J. Shafritz (Ed.), *Classics of public administration* (pp. 396-413). Wadsworth.
- Alonso-Zaldivar, R. (2004). U.S. rethinks air travel screening: Facing questions about privacy issues, the government will try to redesign a computer system to identify suspected terrorists. *Los Angeles Times*, July 16, A20.
- Bajaj, A., & Sudha, R. (2003, October-December). IAIS: A methodology to enable inter-agency information sharing in eGovernment. *Journal of Database Management*, 14(4), 59-80.
- Bond, R., & Whiteley, C. (1998, July). Untangling the Web: A review of certain secure e-commerce legal issues. *International Review of Law, Computers & Technology*, 12(2), 349-370.
- Brown, M.M., & Brudney, J.L. (1998). Public sector information technology initiatives: Implications for programs of public administration. *Administration and Society*, 30(4), 421-442.
- Carey, S., & Power, S. (2003). Responding to privacy concerns, JetBlue e-mails an explanation. *Wall Street Journal*, September 22, B3.
- Dawes, S.S. (2004, January). Training the IT-savvy public manager: Priorities and strategies for public management education. *Journal of Public Affairs Education*, 10(1), 5-17.
- Department of Homeland Security. (2003a, February 13). CAPPS II: Myths and facts.
- Department of Homeland Security. (2003b, July 22). CAPPS II Privacy Act notice. DHS/TSA-2003-1.
- Haque, A. (2001). GIS, public service, and the issue of democratic governance. *Public Administration Review*, 61(3), 259-265.
- Hoffman, I. (2004). Diebold vows to fix e-vote problems. *Oakland Tribune*, March 25.
- Hunter, G.E. (2001). The role of technology in the exercise of voting rights. *Law Technology*, 34(4), 1-14.
- McCartney, S., & Schatz, A. (2004). American released passenger data. *Wall Street Journal*, April 12, A2.
- Milward, H.B., & Provan, K.G. (2000). Governing the hollow state. *Journal of Public Administration Research and Theory*, 10(2), 359-379.
- Niman, M.I. (2004). A brave new world of voting. *The Humanist*, 64(1), 10-13.
- Strickland, L.S. (2003, January/July). Records and information management perspectives, Part I: Legislative and legal developments. *Bulletin of the American Society for Information Science and Technology*, 29(5), 11-15.
- Swartz, N. (2004, January/February). E-government around the world. *Information Management Journal*, 31(1), 12.
- Thomas, J.C., & Streib, G. (2003). The new face of government: Citizen-initiated contacts in the era of e-government. *Journal of Public Administration Research and Theory*, 13(1), 83-102.
- United Nations. (2003). World public sector report 2003. *E-government at the crossroads*. New York: UN Press.
- Whitman, D. (2000). Chadology 101: Divining a dimple. Who knew a simple ballot could be so tricky? *U.S. News & World Report*, 129(21), 34.

## KEY TERMS

**E-Government:** The delivery of local government service through electronic means. The use of technology to improve government operations and create public value.

**Encryption:** A modification process that transforms data into a non-readable format to protect against unauthorized viewing. Encryption can be handled by special applications, but it is often included as a feature of a database system, or as a utility application as part of the operating system. Depending on the encryption/decryption method, information transmitted in encrypted format may be decrypted routinely and without user intervention by e-mail software or commodity Web viewers, products such as Microsoft Internet Explorer, Netscape Navigator, or Mozilla-compliant browsers.

**Geographic Information System (GIS):** A database of a region and software interfaces to view and manage the data. GIS implementation often begins with a digitized map of an area derived from original parcel maps or aerial photography. Multiple “layers” are created for the map to include different infrastructure systems such as roads, sewers, and telecommunications.

**Interface:** The point at which two systems connect, and the method by which communication is accomplished. The computer keyboard, mouse, printer, and video display exemplify interfaces between the machine’s internal operations and the human user.

**Legacy Data:** Contents of databases that precede the installation and implementation of new systems. Optimally, legacy data is migrated into new data systems; following this process, the older application and data structure may be archived or deleted. Frequently, in an effort to reduce the cost of implementation, legacy data remains outside a new data store and accessed as foreign data records from the new application.

**Referential Integrity:** A concept developed as part of relational database management systems. A connecting construct, or “key”, allows a database designer to optimally develop a set of tables while retaining links between related data. With referential integrity, records cannot be updated in isolation in an inconsistent manner.

**Web-Enabling:** A modification process whereby information formerly requiring a locally constrained interface for access becomes available to commodity Web viewers, products such as Microsoft Internet Explorer, Netscape Navigator, or Mozilla-compliant browsers.

# E-Mail Data Stores

**Catherine Horiuchi**  
Seattle University, USA

## INTRODUCTION

*For many people, e-mail has become a running record of their business and personal lives. Somewhere in that big clot of e-mail messages that have accumulated over the years is a wealth of information about people they've met, work they've done, meetings they've held. There are tough calls and tender moments, great debates and funny episodes. When did you first meet a certain person? Just what was the initial offer in a business deal? What was that joke somebody sent you in 1998? The answers lie in your old e-mail. Unfortunately, it's often easier to search the vast reaches of the World Wide Web than to quickly and accurately search your own stored e-mail. (Mossberg, 2004)*

Electronic mail, or e-mail, has evolved from its beginnings as one of the earliest Internet applications. The network originally connected computers to computers, but in 1977, RFC 733 updated the messaging protocol to “focus on people and not mailboxes as recipients” (Crocker, Vittal, Pogran, & Henderson, 1977, p.1). Once considered a simple method to send text messages between two machines, e-mail has become a complex system of hardware and software interfaces between individuals and institutions. Messaging technologies manage the flow of major lines of business and significant public sector policy processes. As a corollary to this, databases associated with e-mail now rank among the most mission-critical data stores in many organizations.

## BACKGROUND

User-oriented client software interfaces create flexibility to map messages in patterns strongly congruent with the way individuals think and organize information. This usability has resulted in e-mail becoming the catch basin of an organization's intellectual capital and institutional history, particularly knowledge and activities that are not captured by software systems focused on basic business processes, such as inventory and accounts receivable. The message store grows organically over time. The central message store is linear in nature, with messages stored chronologically, but copies of those messages also are managed by each message originator and recipi-

ent. Multiple instances of a message are organized in individualistic fashion in multiple locations, based on each user's business and personal priorities. Options also exist to create, send, and store messages in encrypted format. Firms face critical decisions regarding e-mail administration including policies on retention, software package management, and mitigation of risks from worms, viruses, and e-mail bombs. Administering e-mail is further complicated by the multiple parties who have the option to discard, retain, or forward an e-mail message, creating yet more copies: the sender, the recipient, the administrator of the originating mail system, and the administrator of the receiving mail system. Table 1 describes the basic location of e-mail messages.

The highly congruent, highly personal aspects of e-mail have contributed to efforts to capitalize on these attributes in an organized fashion. These efforts have varied in approach, and each faces specific challenges, discussed in the following section.

## DATA MANAGEMENT CHALLENGES

Strategies to capture the knowledge held in e-mails have ranged from benign neglect (e.g., limited to backing up and archiving a central e-mail message store), to direct integration with a primary business application (e.g., using mail message protocols within a supply-chain software package, such as SAP), to sophisticated programming to capitalize on a particular e-mail platform (e.g., business application programming on Lotus Notes). Each approach is complicated by authentication, platform dependence, data corruptibility, and referential integrity issues.

The simplest strategy, benign neglect, is also the most common: The message store is backed up with the rest of the data on the server. If a user inadvertently deletes a message considered important, a request to the system administrator can result in it being restored from backup. This strategy can also meet legal requirements to retain e-mail if the system administrator has been notified of the requirement and has adequate storage capacity and processes in place. However, it is also dependent on the original sender/recipient to reestablish the connection between the particular e-mail message and its context among many issues and correspondents. And if the mes-

Table 1. Where e-mail data is stored

<p>For each e-mail account, a message and any attachments must be stored somewhere until it is deleted. P. Hoffman (2004) offers a succinct list of common component terms:</p> <ul style="list-style-type: none"> <li>• <b>On individual, end-user systems</b>, a common approach for POP3 users. Messages are copied from a mail server message store onto an individual's computer. The messages are then deleted from the server.</li> <li>• <b>On servers</b>, a common approach for IMAP users with web-based mail clients. Messages are stored only on the mail server.</li> </ul> <p>Most mail systems are configured to store messages both on users' machines and in a central database repository. Large message stores are usually included in system backups, resulting in further replication of messages and attachments.</p>
--

sage was encrypted, loss of the original key will result in a need for a brute-force decryption, a time-consuming process. This simplest strategy also fails to address referential integrity, the principle that assures all copies of data are congruent and no part of the data loses its association with other elements. For instance, if an attachment is deleted, the message is incomplete; if an Internet site linked to a message is altered or expires, the message no longer retains the same meaning.

Organizations with multiple hardware and software systems struggle to collect and analyze data at the organizational level (i.e., metadata analysis). To improve connections and reduce the time required for metadata analysis, firms may replace numerous free-standing applications with an enterprise resource management (ERP) package, or combine data sources into a more integrated data warehouse (Zeng, Chang, & Yen, 2003). These ERP and data warehouse solutions include hooks for messaging technologies to establish links between departments and even external companies in supply-chain automation. This linking technology is compatible with several major e-mail engines so that firms can leverage their existing e-mail systems' knowledge in service to these specialized applications.

Combining a corporate-level e-mail system with a corporate-level business software package automates many manual processes and creates a strong audit trail. However, this creates a high degree of dependence on the technology companies who license the software products as well as any consultants who may have been hired to write specialized software routines and database extensions targeting particular business process automation. These type of technology enhancement projects easily run into the tens of millions for initial implementation and millions for annual maintenance. ERP packages and similar integrated software strategies address the referential integrity problem inherent in having multiple systems that describe various aspects of a single transaction. Instead of separate systems that catalog the purchase of a piece of equipment-its installation at a location, maintenance

schedule, depreciation, ultimate removal, and salvaging - a single system tags all these events to the equipment, resulting in a more comprehensive picture. Although this operational cohesion is of high value to management, a firm's dependence on particular vendors results in loss of competitive pressure. Transitioning to an alternate vendor involves major expense, as does changing the integrated system to meet new business requirements. Historically, firms used software/hardware packages for decades, but that was before software tightly programmed employee behaviors that must shift with changing economic cycles and market challenges.

Authentication between major applications and external data stores can be handled in more than one way, with differing security profiles. The least satisfactory method, from a database administrator point of view, assigns administrative privileges to an application. The database administrator does not control rights of users in the application and cannot match data requests to users. It also exposes the data store to hacking exploits designed to enter the data store using other methods, such as a query engine inherent to the database management system. An alternative method, with named users and permissions at both the database and application level, may require users to authenticate multiple times. This basic dilemma underlies efforts for "single sign on." In most instances, an end user has several separate entities established with the operating systems of multiple servers as well as applications, and a portion of their authentication has been established in a pass-through or permissions-table fashion. A firm risks security compromises to information based on the degree to which it maintains an active user directory.

Rather than buying a software package and using the messaging software merely to route transactions, other firms have centered on the messaging itself and written extensive software enhancements to the basic message function. This is the Lotus Notes strategy. It is best suited for firms with substantial intellectual property, as opposed to firms with extensive inventory to track or manu-



facturing processes around which the concept of supply-chain management was originally developed. This strategy uses a technical staff with a deep understanding of both technology and the firm's business. Its principal weakness is the required special programming for each integrated element, resulting in its attention primarily to high-value integration. Lower priority data will remain outside the data store, and any messaging regarding those materials will have no connection to the other systems that manage them.

Customer relationship management (CRM) software, while originating in call-center management, can be designed around multimodal communication with customers. The customer becomes the primary focus, whether contacted by phone, fax, e-mail, standard mail, or direct face-to-face meetings. Information about existing and potential customers is compounded into the data store (Bose & Sugumaran, 2003). This strategy combines the referential integrity of major software systems with the intellectual property management of a Lotus Notes model. Its primary weakness is its origin in telephone technologies and free-standing call-center business models. To the degree that firms contract out noncore operations, they potentially fragment the knowledge or open access to internal systems through very weak authentication protocols to partner firms.

McCray and Gallager (2001) catalogs the following principles for designing, implementing, and maintaining a digital library: Expect change, know your content, involve the right people, be aware of proprietary data rights, automate whenever possible, adhere to standards, and be concerned about persistence. These library principles are equally relevant for designing, implementing, and maintaining an e-mail data store regardless of the knowledge-management strategy adopted. Applications that interact with messaging systems anticipate stability and persistence in the data store. Messages have mixed data types as attachments, users anticipate the messages will be retrievable to the limits of the organization's persistence policy, and they must remain readable despite likely hardware and software changes.

An organization's managers and staff can consider technical options and decide upon a workable information management strategy. This does not end their e-mail data-management considerations. The comprehensive information contained in metadata stores, and the persistence of the information, creates legal and ethical questions that must be considered. Legal matters, new privacy concerns, and historic fiduciary requirements result in a body of intergovernmental regulations to consider as well.

## LEGAL AND ETHICAL QUESTIONS

In the United States, existing e-mail messages are subject to discovery, a legal process to compel the exchange of information. Because of this risk, and to limit the amount of server storage space that must be managed, corporations may establish e-mail policies to routinely purge data stores. To the degree end users transfer messages to their local machines, these efforts can be confounded. Furthermore, e-mail stores are constrained legally to be simultaneously available and unavailable. For public agencies, the Freedom of Information Act requires accessibility, and the Privacy Act of 1974 requires that all personal information be protected (Enneking, 1998).

Business trends and government rulemaking efforts have resulted in a technologically complex environment for e-mail data stores. Globalization—the economic and social changes related to penetration of linking technologies, such as computer networking and the accompanying rise of multinational firms—creates exposure to multiple nation–state rules, though no international body enforces any constraints on global firms; enforcement remains state-specific. A plethora of international regulations on the use of encryption has affected vendors of e-mail technologies and created variations in allowable management of message stores. Case law and federal legislation have tried to reduce the quantity of unsolicited e-mail or “spam” that comprises a sizable percentage of all Internet messages. Regarding data replication, a ruling determined that the state's privacy act was not violated when copies of e-mail messages were printed or stored as evidence (*Washington State v. Townsend*, 2001).

Lawyers have special interests in e-mail in terms of managing legal matters related to technology used by their clients and also within their own firms. Hopkins and Reynolds (2003) argued that ethical representation of a client should require informing the client of the value of an encrypted message, and the use of encryption for all communications, despite a 1999 American Bar Association position that stated lawyers did not violate rules of professional conduct in sending unencrypted e-mail. The capacity to easily encrypt (which is now built into most commercial e-mail products) combined with increases in cyber crime and ease by which data packets can be intercepted, suggests a preference for encrypted e-mail as more secure than traditional mail or faxes that are customary methods for lawyer–client communication.

Inadvertent mistakes in using e-mail applications can result in erroneous transmissions of what in other media might be considered confidential and privileged informa-

tion. For instance, pressing “reply all” instead of “reply” might send information to parties outside a privileged contact (Hall & Estrella, 2000). Judicial and ethical opinion is divided on whether the act of sending the e-mail—even inadvertently—waives attorney-client privilege, resulting in each situation being evaluated individually, according to the jurisdiction involved. For the most part, the receiving party is able to review and retain these materials for use. Table 2 cites instances of these mistakes.

Emerging communication technologies, such as instant messaging and third-party email servers, create new legal challenges for firms trying to manage their information (Juhnke, 2003).

## FUTURE TRENDS

A degree of stability is created by implementing standard records management processes and using application or platform-specific archiving methods for e-mail data stores. But messaging data management is complicated by adoption of new communication media, such as instant messaging (IM) and Webcasting. In IM or chat mode, a user is online in real time with another user at another computer, both network connected. Originally, chat sessions were not captured and stored, but were transient, similar to most telephone calls. And similar to a modern telephone conversation with a customer service representative, IM sessions are now considered a part of the normal business communications to be captured, stored, indexed, and analyzed.

The U.S. government requires financial firms to be able to turn over IM logs within 24 hours of a request, just as with an e-mail. Failing to meet this requirement can be costly: The Securities and Exchange Commission fined five firms \$8.25 million dollars in 2002 for not preserving e-mail and making it available as required (T. Hoffman, 2004).

No straightforward method exists to manage multiple data stores created on separate devices and managed through multiple proprietary networks. Basic individual e-mail stores on computers can generally be sorted into

folders, clustering related ideas on a topic. Store-and-forward mail servers catalog details on recipients of e-mail. Instant messaging is session-oriented and transitory. Although some large, technology-oriented firms store these sessions for replay, they are not so easily indexed and searched, much less integrated into a company’s metadata repository. Rapid adaptation of new communication technologies results in periodic stranding of data and loss of business intelligence.

## CONCLUSION

Over a quarter century has passed since the Internet messaging protocols defining e-mail message transfers were adopted, expanding communication options. E-mail and the enhancement of IM are simple technologies easily mastered and congruent with human psychology. Organizations can capture these interactions, resulting in massive data stores with a rich mix of data types in attachments. Their importance is evident in efforts to incorporate the knowledge collected in these communication media within ERP and CRM installations.

Legal and ethical constraints may exceed the technical and sociologic challenges. To create a framework that can succeed, organizations must include in their business strategies routine assessments of success factors in their adoption and adaptation of proprietary messaging and application software. High failure rates exist for many of these projects. Emerging case law regarding e-mail increases the complexity in maximal use of message stores. Proliferation of messages and limited manageability are side effects of these policies.

These attributes of metadata creation and analysis favor the largest firms with the most sophisticated staff and data-management systems. The smallest, nimblest firms can adopt any emerging trend in information, and must do so to create a strategic niche. Once a trend persists, these small firms can be acquired and incrementally their new data media integrated into the larger firm. So it appears that form and function are indeed merging between technology and process.

Table 2. Most infamous e-mail faux pas

- **Learning the hard way that “DELETE” doesn’t necessarily get rid of e-mail messages.** By 1986, the entire White House was using IBM’s PROFS (“professional office system”) e-mail software. In late November of that year, Oliver North and John Poindexter deleted thousands of e-mail messages as the Iran-Contra scandal broke. The system backups were not destroyed, however, and the Tower Commission was able to obtain this material in its hearings on unauthorized arms-for-hostages deals (Blanton, 1995).
- **The Morris Worm.** The first Internet worm (the Morris worm) was launched November 2, 1988. It exploited a hole in UNIX’s sendmail routine. The spread of the worm was an unintended result of a flaw in Morris’s code (Hafner & Markoff, 1995).

## REFERENCES

- Blanton, T. S., ed. (1995). *White House e-mail: The top-secret messages the Reagan/Bush White House tried to destroy (Book and Disk)*. New York: New Press.
- Bose, R., & Sugumaran, V. (2003). Application of knowledge management technology in customer relationship management. *Knowledge and Process Management*, 10(1), 3-17.
- Crocker, D. H., Vittal, J. J., Pogran, K. T., & Henderson, D. A., Jr. (1977). *Standard for the formation of ARPA network text messages (RFC 733)*. Department of Defense, Defense Advanced Research Projects Agency, Washington, DC.
- Enneking, N. E. (1998). Managing e-mail: Working toward an effective solution. *Records Management Quarterly*, 32(3), 24-38.
- Hafner, K., & Markoff, J. (1995). *Cyberpunk: Outlaws and Hackers on the Computer Frontier (Revised)*. New York: Simon & Schuster.
- Hall, T., J., & Estrella, R. V. (2000). Privilege and the errant email. *The Journal of Proprietary Rights*, 12(4), 2-7.
- Hoffman, P. (2004). Terms used In Internet mail. *Internet Mail Consortium*. Retrieved March 30, 2004, from <http://www.imc.org/terms.html>.
- Hoffman, T. (2004). Banks, brokerages dogged by e-mail regulations. Retrieved June 29, 2004, from <http://www.computerworld.com>
- Hopkins, R. S., & Reynolds, P. R. (2003). Redefining privacy and security in the electronic communication age: A lawyer's ethical duty in the virtual world of the Internet. *The Georgetown Journal of Legal Ethics*, 16(4), 675-692.
- Juhnke, D. H. (2003). Electronic discovery in 2010. *Information Management Journal*, 37(6), 35-42.
- McCray, A. T., & Gallagher, M. E. (2001). Principles for digital library development. *Communications of the ACM*, 44(5), 48-54.
- Mossberg, W. S. (2004, March 25). New program searches hard disks really well, but has rough edges. *Wall Street Journal*, p. B1.
- Washington State v. Townsend, 105 Wn. App. 622, 20 P. 3d. 1027 2001 Wash. App. LEXIS 567 (2001).
- Zeng, Y. Chang., R. H. L., & Yen, D. C. (2003). Enterprise integration with advanced information technologies: ERP

and data warehousing. *Information Management and Computer Security*, 11(2/3), 115-122.

## KEY TERMS

**Attachment:** An extra file, in any file format, that is linked to a e-mail message. The e-mail message itself must structurally conform to messaging protocols.

**E-Mail Bomb:** An attempt to overwhelm a mail server by sending large numbers of e-mails to a particular account, consuming system resources and initiating a denial of legitimate access.

**Header:** The beginning portion of a message. By design, it should contain the source and target address for the message.

**IM:** Instant messaging. A method for real-time communication over a wired or wireless network. An evolution from IRC, inter-relay chat, an early Internet real-time communication protocol.

**IMAP:** Internet message access protocol, defined in RFC 2060.

**Mail Client:** A software process that moves mail from a message store and presents it to a user.

**Mail Server:** A software process that receives mail from other mail systems and manages the message store.

**Message Store:** The physical location where messages are held until a mail client retrieves them. The type of file or files varies by software package. Some are monolithic database structures; some may be encrypted. Others are plain text files.

**POP, POP3:** Post office protocol, defined in RFC 1939. A process that authorizes a transfer of mail messages to a user's computer, then updates the data store source. This update may optionally include deleting the stored message.

**RFCs:** Requests for comments. The process and rules by which distributed computing standards are designed. The RFC process and the Internet's protocols are developed and managed by the Internet Engineering Task Force (IETF; comprehensive information on RFCs can be found online at <http://www.ietf.org/rfc.html>).

**Worm:** A self-replicating, self-propagating program that uses basic computer operating system code to transfer instances of itself from computer to computer.

# Engineering Information Modeling in Databases

**Z.M.Ma**

*Northeastern University, China*

## INTRODUCTION

Computer-based information technologies have been extensively used to help industries manage their processes and information systems become their nervous center. More specifically, databases are designed to support the data storage, processing, and retrieval activities related to data management in information systems. Database management systems provide efficient task support and tremendous gain in productivity is thereby accomplished using these technologies. Database systems are the key to implementing industrial data management. Industrial data management requires database technique support. Industrial applications, however, are typically data- and knowledge-intensive applications and have some unique characteristics (e.g., large volumes of data with complex structures) that makes their management difficult. Product data management supporting various life-cycle aspects in the manufacturing industry, for example, should not only to describe complex product structure but also manage the data of various life-cycle aspects from design, development, manufacturing, and product support. Besides, some new techniques, such as Web-based design and artificial intelligence, have been introduced into industrial applications. The unique characteristics and usage of these new technologies have created many potential requirements for industrial data management, which challenge today's database systems and promote their evolvement.

## BACKGROUND

From a database-technology standpoint, information modeling in databases can be identified at two levels: (conceptual) data modeling and database modeling, which results in conceptual (semantic) data models and logical database models. Generally, a conceptual data model is designed, and then the designed conceptual data model is transformed into a chosen logical database schema. Database systems based on logical database models are used to build information systems for data management. Much attention has been directed at conceptual data modeling of industrial information systems. Product data models, for example, can be viewed as a class

of semantic data models (i.e., conceptual data models) that take into account the needs of engineering data. Recently conceptual data modeling of enterprises has received increased attention.

Generally speaking, traditional ER/EER (Entity-Relationship/Extended Entity Relationship) or UML models in the database area can be used for industrial data modeling at the conceptual level. But limited by the power of the aforementioned data models in industrial data modeling, some new conceptual data models such as IDEF1X and STEP/EXPRESS have been developed. In particular, to implement the share and exchange of industrial data, the Standard for the Exchange of Product Model Data (STEP) is being developed by the International Organization for Standardization (ISO). EXPRESS is the description method of STEP and a conceptual schema language, which can model product design, manufacturing, and production data. EXPRESS model hereby becomes a major one of conceptual data models for industrial data modeling. Much research has been reported on the database implementation of the EXPRESS model in context of STEP, and some software packages and tools are available in markets.

As to industrial data modeling in database systems, the generic logical database models, such as relational, nested relational, and object-oriented databases, have been used. However, these generic logical database models do not always satisfy the requirements of industrial data management. In nontransaction processing, such as CAD/CAM (Computer-Aided Design/Computer-Aided Manufacturing), knowledge-based system, multimedia, and Internet systems, most of the data-intensive application systems suffer from the same limitations of relational databases. Some nontraditional database models based on the aforementioned special, hybrid, or extended database models have been proposed accordingly.

## MAJOR ISSUES AND SOLUTIONS

### Conceptual Data Models

Much attention has been directed at conceptual data modeling of engineering information (Mannisto, Peltonen, Soininen, & Sulonen, 2001; McKay, Bloor, & de



Pennington, 1996). Product data models, for example, can be viewed as a class of semantic data models (i.e., conceptual data models) that take into account the needs of engineering data (Shaw, Bloor, & de Pennington, 1989). Recently, conceptual information modeling of enterprises such as virtual enterprises has received increasing attention (Zhang & Li, 1999). Generally speaking, traditional ER (P. P. Chen, 1976) and EER can be used for engineering information modeling at conceptual level. But, limited by their power in engineering modeling, some new conceptual data models have been developed.

IDEF1X is a method for designing relational databases with a syntax designed to support the semantic constructs necessary in developing a conceptual schema. Some researchers have focused on the IDEF1X methodology (a thorough treatment of the IDEF1X method can be found in Wisdom Systems, 1985). The use of the IDEF1X methodology to build a database for multiple applications was addressed in Kusiak, Letsche, and Zakarian (1997).

As mentioned earlier, STEP provides a means to describe a product model throughout its life cycle and to exchange data between different units. STEP consists of four major categories, namely, *description methods*, *implementation methods*, *conformance testing methodology and framework*, and *standardized application data models/schemata*. EXPRESS (Schenck & Wilson, 1994), being the description methods of STEP and a conceptual schema language, can model product design, manufacturing, and production data, and the EXPRESS model hereby becomes a major conceptual data model for engineering information modeling.

On CAD/CAM development for product modeling, Eastman and Fereshetian (1994) conducted a review and studied five information models used in product modeling, namely, ER, NAIM, IDEF1X, EXPRESS, and EDM. Compared with IDEF1X, EXPRESS can model complex semantics in engineering applications, including engineering objects and their relationships. Based on the EXPRESS model, it is easy to implement the share and exchange of engineering information.

It should be noted that ER/EER, IDEF1X and EXPRESS could model neither knowledge nor fuzzy information. The first effort was undertaken by Zvieli and Chen (1986) to extend ER models to represent three levels of fuzziness. The first level refers to the set of semantic objects, resulting in fuzzy entity sets, fuzzy relationship sets, and fuzzy attribute sets. The second level concerns the occurrences of entities and relationships. The third level relates to the fuzziness in attribute values of entities and relationships. Consequently, ER algebra was fuzzily extended to manipulate fuzzy data. In G. Q. Chen and Kerre (1998), several major notions in the EER model were extended, including fuzzy exten-

sion to generalization/specialization, and shared subclass/category, as well as fuzzy multiple inheritance, fuzzy selective inheritance, and fuzzy inheritance for derived attributes. More recently, using fuzzy sets and possibility distribution (Zadeh, 1978), fuzzy extensions to IDEF1X (Ma, Zhang, & Ma, 2002) and EXPRESS were proposed and (Ma, Zhang, & Ma, 2001; Ma, Zhang, Ma, & Chen, 2000), respectively.

Unified modeling language (UML; Booch, Rumbaugh, & Jacobson, 1998; OMG, 2003), standardized by the Object Management Group (OMG), is a set of OO (object oriented) modeling notations. UML provides a collection of models to capture the many aspects of a software system. From an information modeling point of view, the most relevant model is the class model. The building blocks in this class model are those of classes and relationships. The class model of UML encompasses the concepts used in ER as well as other OO concepts. In addition, it also presents the advantage of being open and extensible, allowing its adaptation to the specific needs of the application, such as workflow modeling of e-commerce (Chang, Chen, Chen, & Chen, 2000) and product structure mapping (Oh, Hana, & Suhb, 2001). In particular, the class model of UML is extended for the representation of class constraints and the introduction of stereotype associations (Mili et al., 2001).

With the popularity of Web-based design, manufacturing and business activities, the requirement has been put on the exchange and share of engineering information over the Web. Because of limitations of HTML in content-based information processing, the World Wide Web Consortium created eXtensible markup language (XML), a language similar in format to HTML, but more extensible. This new language lets information publishers invent their own tags for particular applications or work with other organizations to define shared sets of tags that promote interoperability and that clearly separate content and presentation. XML provides a Web-friendly and well-understood syntax for the exchange of data. Because XML affects the way data is defined and shared on the Web (Seligman & Rosenthal, 2001), XML technology has been increasingly studied, and more and more Web tools and Web servers support XML. Bourret (2001) developed product data markup language, the XML for product data exchange and integration. As to XML modeling at concept level, Conrad, Scheffner and Freytag (2000) used UML was used for designing XML DTD. Xiao, Dillon, Chang, and Feng (2001) developed an object-oriented conceptual model to design XML schema. Lee, Lee, Ling, Dobbie, and Kalinichenko (2001) used the ER model for the conceptual design of semistructured databases in. Note, however, that XML does not support imprecise and uncertain information modeling as well as it does knowledge modeling. Intro-



ducing imprecision and uncertainty into XML has received increased attention (Abiteboul, Segoufin, & Vianu, 2001; Damiani, Oliboni, & Tanca, 2001).

## **LOGICAL DATABASE MODELS**

### **Classical Logical Database Models**

For engineering information modeling in database systems, the generic logical database models (e.g., relational databases, nested relational databases, object-oriented databases) can be used. Also, some hybrid logical database models, such as object-relational databases, are very useful for this purpose. On top of a relational DBMS, Arnalte and Scala (1997) build an EXPRESS-oriented information system for supporting information integration in a computer-integrated manufacturing environment. In this case, the conceptual model of the information was built in EXPRESS and then parsed and translated to the corresponding relational constructs. Relational databases for STEP/EXPRESS were also discussed in Krebs and Lühsen (1995). In addition, Barsalou and Wiederhold (1990) developed an object-oriented layer to model complex entities on top of a relational database. This domain-independent architecture permits object-oriented access to information stored in relational format information that can be shared among applications.

Object-oriented databases provide an approach for expressing and manipulating complex objects. A prototype object-oriented database system called ORION was thus designed and implemented to support CAD (Kim, Banerjee, Chou, & Garza, 1990). Goh, Je, Song, and Wang (1994, 1997) studied object-oriented databases for STEP/EXPRESS. In addition, Dong, Chee, He, and Goh (1997) designed an object-oriented active database for STEP/EXPRESS models. According to the characteristics of engineering design, Samaras, Spooner, and Hardwick (1994) provided a framework for the classification of queries in object-oriented engineering databases, in which the strategy for query evaluation is different from traditional relational databases.

### **XML Databases**

It is crucial for Web-based applications to model, store, manipulate, and manage XML data documents. XML documents can be classified into data-centric documents and document-centric documents (Bourret, 2001). Data-centric documents have a fairly regular structure, fine-grained data (i.e., the smallest independent unit of

data is at the level of a PCDATA-only element or an attribute), and little or no mixed content. The order in which sibling elements and PCDATA occurs is generally not significant, except when validating the document. Data-centric documents are documents that use XML as a data transport. They are designed for machine consumption, and it is usually superfluous that XML is used at all. That is, it is not important to the application or the database that the data is, for some length of time, stored in an XML document.

As a general rule, the data in data-centric documents is stored in a traditional database, such as a relational, object-oriented, or hierarchical database. The data can also be transferred from a database to an XML document. For the transfers between XML documents and databases, the mapping relationships between their architectures and their data should be created (Lee & Chu, 2000; Surjanto, Ritter, & Loeser, 2000). Note that it is possible to discard information such as the document and its physical structure when transferring data between them. It must be pointed out, however, that data in data-centric documents, such as semistructured data, can also be stored in a native XML database, in which a document-centric document is usually stored. Document-centric documents are characterized by less regular or irregular structure, larger grained data (i.e., the smallest independent unit of data might be at the level of an element with mixed content or the entire document itself), and a large amount of mixed content. The order in which sibling elements and PCDATA occur is almost always significant. Document-centric documents are usually designed for human consumption. As a general rule, document-centric documents are stored in a native XML database or a content management system (i.e., an application designed to manage documents and built on top of a native XML database). Native XML databases are databases designed especially for storing XML documents. The only difference between native XML databases and other databases is that their internal model is based on XML and not something else, such as the relational model.

In practice, however, the distinction between data-centric and document-centric documents is not always clear. So the aforementioned rules are not absolute. Data, especially semistructured data, can be stored in native XML databases, and documents can be stored in traditional databases when few XML-specific features are needed. Furthermore, the boundaries between traditional databases and native XML databases are beginning to blur, as traditional databases add native XML capabilities and native XML databases support the storage of document fragments in external databases.

## Special, Hybrid and Extended Logical Database Models

It should be pointed out that the generic logical database models, such as relational databases, nested relational databases, and object-oriented databases, do not always satisfy the requirements of engineering modeling. As pointed out in Liu (1999), relational databases do not describe the complex structure relationship of data naturally, and separate relations may result in data inconsistencies when updating the data. In addition, the problem of inconsistent data still exists in nested relational databases, and the mechanism of sharing and reusing CAD objects is not fully effective in object-oriented databases. In particular, these database models cannot handle engineering knowledge. Special databases, such as those based on relational or object-oriented models, are hereby introduced. Dong and Goh (1998) developed an object-oriented active database for engineering application to support intelligent activities in engineering applications. Deductive databases were considered preferable to database models for CAD databases and deductive object-relational databases for CAD were introduced (Liu, 1999). Constraint databases based on the generic logical database models are used to represent large or even infinite sets in a compact way and are suitable hereby for modeling spatial and temporal data (Belussi, Bertino, & Catania, 1998; Kuper, Libkin, & Paredaens, 2000). Also, it is well established that engineering design is a constraint-based activity (Dzbor, 1999; Guiffrida & Nagi, 1998; Young, Giachetti, & Ress, 1996). So constraint databases are promising as a technology for modeling engineering information that can be characterized by large data in volume, complex relationships (i.e., structure, spatial, or temporal semantics), and intensive knowledge. Posselt and Hillebrand (2002) investigated the issue about constraint database support for evolving data in product design.

It should be noticed that fuzzy databases have been proposed to capture fuzzy information in engineering (Sebastian & Antonsson, 1996; Zimmermann, 1999). Fuzzy databases may be based on the generic logical database models such as relational databases (Buckles & Petry, 1982; Prade & Testemale, 1984), nested relational databases (Yazici, Soysal, Buckels, & Petry, 1999), and object-oriented databases (Bordogna, Pasi, & Lucarella, 1999; George, Srikanth, Petry, & Buckles, 1996; Van Gyseghem & Caluwe, 1998). Also, some special databases are extended for fuzzy information handling. Medina, Pons, Cubero, and Vila, (1997) presented the architecture for deductive fuzzy relational database, and Bostan and Yazici (1998) proposed a fuzzy deductive object-oriented data model. More recently, Saygin and Ulusoy (2001) investigated how to construct

fuzzy event sets automatically and to apply them to active databases.

## FUTURE TRENDS

Database modeling for engineering serves as engineering applications, and the evolution of engineering application in data management challenges database technologies and promotes their evolvement. Database modeling must provide effective support for current Web-based, distributed, knowledge-based, and intelligent engineering data management.

## CONCLUSION

Computer, network, and information technologies have been the foundation of today's industrial enterprises. Enterprises obtain an increasing variety of products and products with low prices, high quality, and shorter lead time by using these technologies. On the other hand, as manufacturing engineering is a data and knowledge intensive application area, some new requirements are needed, which impose a challenge to current technologies and promote their evolvement. Database systems, being the depository of data, are the cores of information systems and provide the facilities of data modeling and data manipulation.

This article reviews database technologies in engineering information modeling. The contribution of the article is to identify the direction of database study viewed from engineering applications and provide modeling tools for engineering design, manufacturing, and production management. Perhaps more powerful database models will be developed to satisfy engineering information modeling.

## REFERENCES

- Abiteboul, S., Segoufin, L., & Vianu, V. (2001). Representing and querying XML with incomplete information. *Proceedings of the 12th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (pp. 150-161).
- Arnalte, S., & Scala, R. M. (1997). An information system for computer-integrated manufacturing systems. *Robotics and Computer-Integrated Manufacturing*, 13(3), 217-228.
- Barsalou, T., & Wiederhold, G. (1990). Complex objects for relational databases. *Computer-Aided Design*, 22(8), 458-468.

- Belussi, A., Bertino, E., & Catania, B. (1998). An extended algebra for constraint databases. *IEEE Transactions on Knowledge and Data Engineering*, 10(5), 686-705.
- Booch, G., Rumbaugh, J., & Jacobson, I. (1998). *The unified modeling language user guide*. Essex, UK: Addison-Wesley Longman.
- Bordogna, G., Pasi, G., & Lucarella, D. (1999). A fuzzy object-oriented data model for managing vague and uncertain information. *International Journal of Intelligent Systems*, 14, 623-651.
- Bostan, B., & Yazici, A. (1998). A fuzzy deductive object-oriented data model. *Proceedings of the IEEE International Conference on Fuzzy Systems* (Vol. 2, pp. 1361-1366).
- Bourret, R. (2001). *XML and databases*. Retrieved from <http://www.rpbourret.com/xml/XMLAndDatabases.htm>
- Buckles, B. P., & Petry, F. E. (1982). A fuzzy representation of data for relational database. *Fuzzy Sets and Systems*, 7(3), 213-226.
- Chang, Y. L., Chen, S., Chen, C. C., & Chen, I. (2000). Workflow process definition and their applications in e-commerce. *Proceedings of International Symposium on Multimedia Software Engineering* (pp. 193-200).
- Chen, G. Q., & Kerre, E. E. (1998). Extending ER/EER concepts towards fuzzy conceptual data modeling. *Proceedings of the 1998 IEEE International Conference on Fuzzy Systems* (Vol. 2, pp. 1320-1325).
- Chen, P. P. (1976). The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems*, 1(1), 9-36.
- Conrad, R., Scheffner, D., & Freytag, J. C. (2000). XML conceptual modeling using UML. *Lecture Notes in Computer Science*, 1920, 558-571.
- Damiani, E., Oliboni, B., & Tanca, L. (2001). Fuzzy techniques for XML data smushing. *Lecture Notes in Computer Science*, 2206, 637-652.
- Dong, Y., & Goh, A. (1998). An intelligent database for engineering applications. *Artificial Intelligence in Engineering*, 12, 1-14.
- Dong, Y., Chee, C. F. Y., He, Y., & Goh, A. (1997). Active database support for STEP/EXPRESS models. *Journal of Intelligent Manufacturing*, 8(4), 251-261.
- Dzbor, M. (1999). Intelligent support for problem formalisation in design. *Proceedings of the 3rd IEEE Conference on Intelligent Engineering Systems*.
- Eastman, C. M., & Fereshetian, N. (1994). Information models for use in product design: A comparison. *Computer-Aided Design*, 26(7), 551-572.
- George, R., Srikanth, R., Petry, F. E., & Buckles, B. P. (1996). Uncertainty management issues in the object-oriented data model. *IEEE Transactions on Fuzzy Systems*, 4(2), 179-192.
- Goh, A., Hui, S. C., Song, B., & Wang, F. Y. (1994). A study of SDAI implementation on object-oriented databases. *Computer Standards & Interfaces*, 16, 33-43.
- Goh, A., Hui, S. C., Song, B., & Wang, F. Y. (1997). A STEP/EXPRESS to object-oriented databases translator. *International Journal of Computer applications in Technology*, 10(1/2), 90-96.
- Guiffrida, A., & Nagi, R. (1998). Fuzzy set theory applications in production management research: A literature survey. *Journal of Intelligent Manufacturing*, 9, 39-56.
- ISO IS 10303-1 TC184/SC4: Product Data Representation and Exchange-Part 1: Overview and Fundamental Principles, International Standard, 1994.
- ISO IS 10303-1 TC184/SC4: Product Data Representation and Exchange-Part 11: The EXPRESS Language Reference Manual, International Standard, 1994.
- Kim, W., Banerjee, J., Chou, H. T., & Garza, J. F. (1990). Object-oriented database support for CAD. *Computer-Aided Design*, 22(8), 521-550.
- Krebs, T., & Lührsen, H. (1995). STEP databases as integration platform for concurrent engineering. *Proceedings of the 2nd International Conference on Concurrent Engineering* (pp. 131-142).
- Kuper, G., Libkin, L., & Paredaens, J. (2000). *Constraint databases*. Heidelberg, Germany: Springer-Verlag.
- Kusiak, A., Letsche, T., & Zakarian, A. (1997). Data modeling with IDEF1X. *International Journal of Computer Integrated Manufacturing*, 10(6), 470-486.
- Lee, D. W., & Chu, W. W. (2000). Constraints-preserving transformation from XML document type definition to relational schema. *Lecture Notes in Computer Science*, 1920, 323-338.
- Lee, M. L., Lee, S. Y., Ling, T. W., Dobbie, G., & Kalinichenko, L. A. (2001). Designing semistructured databases: A conceptual approach. *Lecture Notes in Computer Science*, 2113, 12-21.
- Liu, M. C. (1999). On CAD databases. *Proceedings of the 1999 IEEE Canadian Conference on Electrical and Computer Engineering* (pp. 325-330).

- Ma, Z. M. (2004). Imprecise and uncertain engineering information modeling in databases: Models and formal transformations. In P. van Bommel (Ed), *Transformation of knowledge, information and data: Theory and applications* (pp. 153-175). Hershey, PA: Infosci.
- Ma, Z. M., Lu, S. Y., & Fotouhi, F. (2003). Conceptual data models for engineering information modeling and formal transformation of EER and EXPRESS-G. *Lecture Notes in Computer Science*, 2813, 573-75.
- Ma, Z. M., Zhang, W. J. & Ma, W. Y. (2001). Fuzzy data type modeling with EXPRESS. *Proceedings of the ASME 2001 International Design Engineering Technical Conferences and the Computers and Information in Engineering Conference*, September 28-October 2, Salt Lake City, Utah.
- Ma, Z. M., Zhang, W. J., & Ma, W. Y. (2002). Extending IDEF1X to model fuzzy data. *Journal of Intelligent Manufacturing*, 13(4), 295-307.
- Ma, Z. M., Zhang, W. J., Ma, W. Y., & Chen, G. Q. (2000). Extending EXPRESS-G to model fuzzy information in product data model. *Proceedings of the 2000 ASME Design Engineering Technical Conference*.
- Mannisto, T., Peltonen, H., Soininen, T. & Sulonen, R. (2001). Multiple abstraction levels in modeling product structures. *Date and Knowledge Engineering*, 36(1), 55-78.
- McKay, A., Bloor, M. S., & de Pennington, A. (1996). A framework for product data. *IEEE Transactions on Knowledge and Data Engineering*, 8(5), 825-837.
- Medina, J. M., Pons, O., Cubero, J. C., & Vila, M. A. (1997). FREDDI: A fuzzy relational deductive database interface. *International Journal of Intelligent Systems*, 12(8), 597-613.
- Mili, F., Shen, W., Martinez, I., Noel, Ph., Ram, M., & Zouras, E. (2001). Knowledge modeling for design decisions. *Artificial Intelligence in Engineering*, 15, 153-164.
- Oh, Y., Hana, S. H., & Suhb, H. (2001). Mapping product structures between CAD and PDM systems using UML. *Computer-Aided Design*, 33, 521-529.
- OMG. (2003). Unified modeling language (UML). Retrieved from <http://www.omg.org/technology/documents/formal/uml.htm>
- Posselt, D., & Hillebrand, G. (2002). Database support for evolving data in product design. *Computers in Industry*, 48(1), 59-69.
- Prade, H., & Testemale, C. (1984). Generalizing database relational algebra for the treatment of incomplete or uncertain information. *Information Sciences*, 34, 115-143.
- Samaras, G., Spooner, D., & Hardwick, M. (1994). Query classification in object-oriented engineering design systems. *Computer-Aided Design*, 26(2), 127-136.
- Saygin, Y., & Ulusoy, O. (2001). Automated construction of fuzzy event sets and its application to active databases. *IEEE Transactions on Fuzzy Systems*, 9(3), 450-460.
- Schenck, D. A., & Wilson, P. R. (1994). *Information modeling: The EXPRESS way*. Cambridge, UK: Oxford University Press.
- Sebastian, H. J., & Antonsson, E. K. (1996). *Fuzzy sets in engineering design and configuration*. Norwell, MA: Kluwer Academic Publishers.
- Seligman, L., & Rosenthal, A. (2001, June). XML's impact on databases and data sharing. *IEEE Computer*, 34(6), 59-67.
- Shaw, N. K., Bloor, M. S., & de Pennington, A. (1989). Product data models. *Research in Engineering Design*, 1, 43-50.
- Surjanto, B., Ritter, N., & Loeser, H. (2000). XML content management based on object-relational database technology. *Proceedings of the First International Conference on Web Information Systems Engineering* (Vol. 1, pp. 70-79).
- Van Gysegheem, N., & Caluwe, R. D. (1998). Imprecision and uncertainty in UFO database model. *Journal of the American Society for Information Science*, 49(3), 236-252.
- Wisdom Systems. (1985). *U.S. Air Force ICAM manual: IDEF1X*. Naperville, IL.
- Xiao, R. G., Dillon, T. S., Chang, E., & Feng, L. (2001). Modeling and transformation of object-oriented conceptual models into XML schema. *Lecture Notes in Computer Science*, 2113, 795-804.
- Yazici, A., Soysal, A., Buckles, B. P., & Petry, F. E. (1999). Uncertainty in a nested relational database model. *Data & Knowledge Engineering*, 30, 275-301.
- Young, R. E., Giachetti, R., & Ress, D. A. (1996). A fuzzy constraint satisfaction system for design and manufacturing. *Proceedings of the 5th IEEE International Conference on Fuzzy Systems* (Vol. 2, pp. 1106-1112).



Zadeh, L. A. (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1(1), 3-28.

Zhang, W. J., & Li, Q. (1999). Information modeling for made-to-order virtual enterprise manufacturing systems. *Computer-Aided Design*, 31(10), 611-619.

Zimmermann, H. J. (1999). *Practical Applications of Fuzzy Technologies*. Norwell, MA: Kluwer Academic Publishers.

Zvieli, A., & Chen, P. P. (1986). Entity-relationship modeling and fuzzy databases. *Proceedings of the 1986 IEEE International Conference on Data Engineering*, 320-327.

## KEY TERMS

**Conceptual Data Modeling of Engineering Information:** Using conceptual data models to implement the data modeling of engineering information. The conceptual data models for engineering data modeling include some special conceptual data models for industry, such as EXPRESS/STEP and IDEF1X, and some traditional conceptual data models, such as ER/EER and UML.

**Data Modeling:** Implementing data management in engineering information systems with information technology and, in particular, database technology. The complex data semantics and semantic relationships are described in data modeling.

**Database Models:** Conceptual data models and logical database models.

**Engineering Information Systems:** The information systems used to manage the information in data and knowledge-intensive engineering applications and to implement various engineering activities.

**EXPRESS/STEP:** To share and exchange product data, the Standard for the Exchange of Product Model Data (STEP) is being developed by the International Organization for Standardization (ISO). EXPRESS is the description methods of STEP and can be used to model product design, manufacturing, and production data.

**IDEF1X:** IDEF1X, one of IDEF tools developed by Integrated Computer-Aided Manufacturing (ICAM), is a formal framework for consistent modeling of the data necessary for the integration of various functional areas in computer integrated manufacturing (CIM).

**Logical Database Modeling of Engineering Information:** Using logic database models to implement the data modeling of engineering information. The logical database models for engineering data modeling include some traditional database models such as relational database model and object-oriented database model and some special, hybrid, and extended database models.



# Ensuring Serializability for Mobile–Client Data Caching

**Shin Parker**

*University of Nebraska at Omaha, USA*

**Zhengxin Chen**

*University of Nebraska at Omaha, USA*

## IMPORTANCE OF ENSURING SERIALIZABILITY IN MOBILE ENVIRONMENTS

Data management in mobile computing has emerged as a major research area, and it has found many applications. This research has produced interesting results in areas such as data dissemination over limited bandwidth channels, location-dependent querying of data, and advanced interfaces for mobile computers (Barbara, 1999). However, handling multimedia objects in mobile environments faces numerous challenges. Traditional methods developed for transaction processing (Silberschatz, Korth & Sudarshan, 2001) such as concurrency control and recovery mechanisms may no longer work correctly in mobile environments. To illustrate the important aspects that need to be considered and provide a solution for these important yet “tricky” issues in this article, we focus on an important topic of data management in mobile computing, which is concerned with how to ensure serializability for mobile-client data caching. New solutions are needed in dealing with caching multimedia data for mobile clients, for example, a cooperative cache architecture was proposed in Lau, Kumar, and Vankatesh (2002). The particular aspect considered in this article is that when managing a large number of multimedia objects within mobile client-server computing environments, there may be multiple physical copies of the same data object in client caches with the server as the primary owner of all data objects. Invalid-access prevention policy protocols developed in traditional DBMS environment will not work correctly in the new environment, thus, have to be extended to ensure that the serializability involving data updates is achieved in mobile environments. The research by Parker and Chen (2004) performed the analysis, proposed three extended protocols, and conducted experimental studies under the invalid-access prevention policy in mobile environments to meet the serializability requirement in a mobile client/server environment that deals with multimedia objects. These three protocols, referred to as extended server-based two-phase locking (ES2PL), extended call back

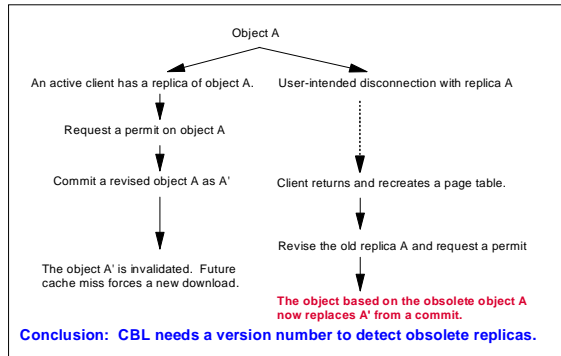
locking (ECBL), and extended optimistic two-phase locking (EO2PL) protocols, have included additional attributes to ensure multimedia object serializability in mobile client/server computing environments. In this article, we examine this issue, present key ideas behind the solution, and discuss related issues in a broader context.

## BACKGROUND

In a typical client-server computing architecture, there may exist multiple physical copies of the same data object at the same time in the network with the server as the primary owner of all data objects. The existence of multiple copies of the same multimedia object in client caches is possible when there is no data conflict in the network. In managing multiple clients’ concurrent read/write operations on a multimedia object, no transactions that accessed the old version should be allowed to commit. This is the basis of the invalid-access prevention policy, from which several protocols have been proposed. The purpose of these protocols is to create an illusion of a single, logical, multimedia data object in the face of multiple physical copies in the client/server network when a data conflict situation arises. When the server becomes aware of a network-wide data conflict, it initiates a cache consistency request to remote clients on behalf of the transaction that caused the data conflict. Three well-known invalid-access prevention protocols are Server-based Two-Phase Locking (S2PL), Call-Back Locking (CBL), and Optimistic Two-Phase Locking (O2PL).

In order to extend these policies to the mobile environment, we should understand that there are four key constraints of mobility which forced the development of specialized techniques, namely, unpredictable variation in network quality, lowered trust and robustness of mobile elements, limitations on local resources imposed by weight and size constraints, and concern for battery power consumption (Satyanarayanan, 1996). The inherent limitations of mobile computing systems present a challenge to the traditional problems of database management, especially when the client/server communication is

Figure 1. CBL failure analysis tree in a mobile environment



unexpectedly severed from the client site. The standard policy does not enforce the serializability to the mobile computing environment. Transactions executing under an avoidance-based scheme must obey the Read-Once Write-All (ROWA) principle, which guarantees the correctness of the data from the client cache under the CBL or the O2PL protocol. The standard CBL and O2PL protocols cannot guarantee the currency of the mobile clients' cache copies or prevent serializability violations when they reconnect to the network. Figures 1 illustrates how error conditions (appearing toward the end of the figure) arise after mobile clients properly exit the client application when the traditional CBL protocol is used.

## FUNDAMENTAL ISSUES AND APPROACHES TO DEALING WITH THESE ISSUES

In order to extend invalid-access prevention policy protocols to mobile environments, there are three fundamental issues that need to be addressed for mobile-client multimedia data caching, namely:

- to transform multimedia objects from databases' persistent data type to the clients' persistent data type;
- to handle client-server communication for multimedia objects; and
- to deal with the impact of mobility, particularly to deal with the case when the client-server communication is unexpectedly severed from the client site.

Research work from various authors (Breitbart et al., 1999; Franklin, Carey & Livny, 1997; Jensen & Lomer, 2001; Pacitti, Minet & Simon, 1999; Shanmugasundaram et al., 1999; Schuldt, 2001) have contributed to the inves-

Table 1. Extended invalid-access prevention policy

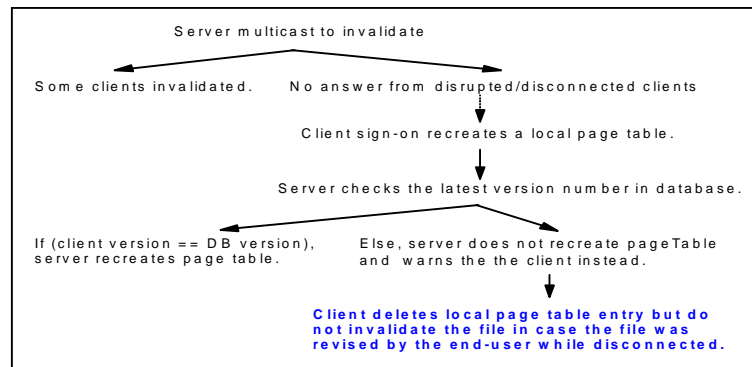
ATTRIBUTE	S2PL	O2PL	CBL
Version Numbers	X	<b>X</b>	<b>X</b>
Recreate/release page table rows		<b>X</b>	<b>X</b>
Permit before Commit			X
Lock before Commit	X		
Commit before Lock		X	
Invalidation		X	X
Dynamic Replication	X	X	
Broadcast		X	X
Read-write conflict		X	X
Write-read conflict	X	X	X
Write-write conflict	X	X	
Relinquish unused locks at sign-off	<b>X</b>	<b>X</b>	<b>X</b>
Maximum lock duration	<b>X</b>	<b>X</b>	<b>X</b>
Server knows who has locks	X	X	X
Server knows who has what objects		X	X

tigation of aspects related to ensuring serializability of data management. Based on these studies, Parker and Chen (2004) have conducted a more recent research to deal with the three issues mentioned above and developed algorithms to achieve extended invalid-access prevention protocols. The basic ideas of this research are summarized below.

First, in order to prevent the serializability failure scenario described above, we summarize important features of the extended invalid-access prevention policy protocols for the mobile client/server environments that guarantee the serializability. As shown in Table 1, an X denotes an attribute under the standard invalid-access prevention policy, while a bold-face **X** as an additional attribute under the extended invalid-access prevention policy. The revised algorithms for extended invalid-access prevention policy protocols are developed based on these considerations.

As an example of these attributes, here we take a brief look at the important role of the page table. To detect or avoid invalid-accesses from all transactions, all clients and the server each need to keep a separate table to detect or avoid data conflict situations. For clients, page tables are the current inventories of their cached multimedia objects. For the server, a page table is the information

Figure 2. Consistency check of the page table



about their active constituent clients to detect or avoid data conflicts in the network. Figure 2 depicts a proper page table procedure for logical invalidations to deal with serializability problems through page table consistency check.

## EXTENDING TRADITIONAL PROTOCOLS: BASIC IDEA AND RESULTS OF EXPERIMENTS

To illustrate the basic ideas involved in extending standard protocols, let us take a look at the case of extended O2PL algorithm with dynamic replication. In its original form, the O2PL protocol defers the write declaration until the end of a transaction’s execution phase. Dynamic replication is a natural choice for the O2PL because when the server issues a write lock after multicasting to all cached remote clients of the same object, the server already has an updated object at hand from the committing client. Just before the server commits the new object to the database with an exclusive lock, there are two copies of the new version object in the network, the server’s binary array variable and the local client’s new-version cache copy and only one primary copy of the old version object in the database.

The correctness criterion in the replicated database is one-copy serializability (Holiday, Agrawal & Abadi, 2002) under the ROWA principle where write operations must write to all copies before the transaction can complete. This is accomplished via the server’s multicast transmission to a subset of active clients after the primary new-version copy is safely stored in the database.

The primary copy of an object is with the server, and the replicas are at client caches for read transactions. For write transactions, the primary copy is at the transaction’s

originating site temporarily. After commit operations, however, the server becomes the primary site, and the originating client becomes one of the replicated sites. The server then multicasts the replica to remote clients with the previous version object. When a client downloads an object explicitly, a local lock is given automatically, but the end user can release the local lock manually. Local locks will not be automatically given after dynamic replications.

To enforce the network-wide unique object ID, the server application will verify the uniqueness of the file name at the insert transaction, and the client application will verify that the file name is not altered at the commit transaction as an early abort step.

## RESULT OF EXPERIMENTS ON EXTENDED PROTOCOLS

The three extended protocols have been implemented, and comparative studies through experiments have been conducted. Below is the result of an experiment where identical transactions of four clients are used, each with two multimedia objects (one small size and the other 15 times as large). Table 2 summarizes the number of messages clients sent to the server, total kilobytes of the messages, the number of the server aborts, and the abort rate which is the percentage of aborts from the entire number of messages clients sent to the server. Any dynamic replications do not count toward the messages sent since clients do not send any messages to the server for them. All client-initiated abort messages, such as the permit abort in the ECBL or the lock abort in the ES2PL, are counted toward the MESSAGE, not the ABORT.

Experiments have shown that extended invalid-access prevention policy algorithms enforce a guaranteed serializability of multimedia objects in RDBMS applications under a mobile client/server environment. As for



Table 2. Comparison of extended invalid-access prevention policy protocols

PROTOCOL	MSSG Nr	KB*	ABORT	ABORT RATE
ES2PL-Replication	18	70	3	17%
ECBL-Invalidation	34	44	2	6%
EO2PL-Rep+Inval	30	42	0	0%
EO2PL-Replication	24	41	0	0%

the pros and cons of each extended algorithm, we have the following general observations. Extended S2PL protocol brings the lowest number of client messages to the server but at the highest server abort rate leaving the network with multiple versions. Extended CBL protocol with invalidation carries the highest number of client messages sent to the server and a moderate server abort rate in the expense of reliability. Extended O2PL protocol with replication offers a moderate number of client messages sent to the server with the lowest server abort rate that may make it desirable for most applications.

## FUTURE TRENDS

Due to its importance for data management in a mobile environment, techniques for ensuring serializability in dealing with multiple copies of multimedia objects should be further explored. New solutions are needed for existing problems in new environments, and new problems emerge, demanding solutions, as well. In this article, we have deliberately focused on a well-selected specific topic to show the need for an in-depth study of dealing with mobility in databases involving multimedia objects. However, the methodology used here can be extended to many other topics in regard to data management in a mobile computing environment. The key to success in such studies lies in a good understanding of important features of mobile environments, as well as inherent limitations of involved resources in such environments.

In addition, ensuring serializability has implications beyond guaranteeing correctness of transaction execution in a mobile environment. For example, based on the work reported above, Parker, Chen, and Sheng (2004) further identified four core areas of issues to be studied in database-centered mobile data mining, with an emphasis on issues related to DBMS implementation such as query processing and transaction processing. Other aspects related to data mining techniques and distributed or mobile (or even pervasive) computing environments have

also been explored (e.g., Kargupta & Joshi, 2001; Lim et al., 2003; Liu, Kargupta & Ryan, 2004; Saygin & Ulusoy, 2002).

There are many other database issues related to mobile computing, such as location-dependent queries (Dunham, Helal & Balakrishnan, 1997; Seydim, Dunham & Kumar, 2001; Yan, Chen & Zhu, 2001). In addition, many issues related to mobile computing can be examined in a more general context of pervasive computing (or ubiquitous computing). Satyanarayanan (1996, 2001) discussed several important issues and future directions on pervasive computing. A wide range of data management aspects should be explored in the future, with the following as sample topics:

- Infrastructure for mobile computing research
- User interface management for pervasive devices
- Data models for mobile information systems
- Mobile database management and mobility-aware data servers
- Mobile transaction and workflow management and models
- Data and process migration, replication/caching and recovery
- Moving objects and location-aware data management
- Adaptability and stability of pervasive systems in ever-changing wireless environments
- Quality of service (QOS) mechanism for mobile data management

## CONCLUSION

As noted earlier, handling multimedia objects in mobile environments faces numerous challenges. Traditional methods developed for transaction processing such as concurrency control and recovery mechanisms may no longer work correctly in mobile environments. In this article, we have focused on a specific issue to ensure

serializability for mobile-client data caching. We have explained why the traditional approaches need to be revised and demonstrated the basic idea of extended approaches. Extending from this particular study, we have also discussed related issues in a more general perspective. As indicated in the Future Trends section, there are numerous challenging issues to be resolved in the near future.

## REFERENCES

- Barbara, D. (1999). Mobile computing and databases: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 11(1), 108-117.
- Breitbart, Y., Komondoor, R., Rastogi, R., Seshadri, S., & Silberschatz, A. (1999). Update propagation protocols for replicated databases. *Proceedings of the 1999 ACM SIGMOD Conference* (pp. 97-108).
- Dunham, M.H., Helal, A., & Balakrishnan, T. (1997). Mobile transaction model that captures both the data and movement behavior. *Mobile Networks and Applications*, 2(2), 149-162.
- Franklin, M.J., Carey, M.J., & Livny, M. (1997). Transactional client-server cache consistency: Alternatives and performance. *ACM Transactions on Database Systems*, 22(3), 315-363.
- Holiday, J., Agrawal, D., & Abbadi, A. (2002). Disconnection modes for mobile databases. *Wireless Networks*, 8(4), 391-402.
- Jensen, C.S., & Lomer, D.B. (2001). Transaction timestamping in temporal databases. *Proceedings of the 27th International Conference on Very Large Data Bases* (pp. 441-450).
- Kargupta, H., & Joshi, A. (2001). Data mining "to go": Ubiquitous KDD for mobile and distributed environments. *Tutorial Notes of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 4.1-4.78).
- Lau, W.H.O., Kumar, M., & Vankatesh, S. (2002). A cooperative cache architecture in support of caching multimedia objects in MANETs. *Proceedings of the WoMMoM 02* (pp. 56-63).
- Lim, E.-P., Wang, Y., Ong, K.-L., & Hwang, S.-Y. (2003, July). In search of knowledge about mobile users. Center for Advanced Information Systems at Nanyang Technological University, Singapore. Retrieved February 5, 2005, from [http://www.ercim.org/publication/Ercim\\_News/enw54/lim.html](http://www.ercim.org/publication/Ercim_News/enw54/lim.html)
- Liu, K., Kargupta, H., & Ryan, J. (2004, January). Distributed data mining bibliography. University of Maryland Baltimore County. Retrieved February 5, 2005, from <http://www.cs.umbc.edu/~hillol/DDMBIB/ddmbib.pdf>
- Pacitti, E., Minet, P., & Simon, E. (1999). Fast algorithms for maintaining replica consistency in lazy master replicated databases. *Proceedings of the 25th International Conference on Very Large Data Bases* (pp. 126-137).
- Parker, S., Chen, Z., & Sheng, E. (2004). Ensuring serializability for mobile data mining on multimedia objects. *Proceedings of the CASDMKM 2004* (pp. 90-98).
- Parker, S., & Chen, Z. (2004). Extending invalid-access prevention policy protocols for mobile-client data caching. *Proceedings of the ACM SAC 2004* (pp. 1171-1176).
- Satyanarayanan, M. (1996). Mobile information access. *IEEE Personal Communications*, 3(1), 26-33.
- Satyanarayanan, M. (2001). Pervasive computing: Vision and challenges. *IEEE Personal Communications*, 8, 10-17.
- Saygin, Y., & Ulusoy, Ö. (2002). Exploiting data mining techniques for broadcasting data in mobile computing environments. *IEEE Transactions on Knowledge Data Engineering*, 14(6), 1387-1399.
- Schuldt, H. (2001). Process locking: A protocol based on ordered shared locks for the execution of transactional processes. *Proceedings of the 20th ACM SIGMOD SIGACT SIGART Symposium on Principles of Database Systems* (pp. 289-300).
- Seydim, A.Y., Dunham, M.H., & Kumar, V. (2001). Location dependent query processing. *Proceedings of the 2nd ACM International Workshop on Data Engineering for Wireless and Mobile Access* (pp. 47-53).
- Shanmugasundaram, S., Nithrakashyap, A., Sivasankaran, R., & Ramamritham, K. (1999). Efficient concurrency control for broadcast environments. *Proceedings of the 1999 ACM SIGMOD International Conference in Management of Data* (pp. 85-96).
- Silberschatz, A., Korth, H.F., & Sudarshan, S. (2001). *Database system concepts* (4th ed.). New York: WCB McGraw-Hill.
- Yan, J., Chen, Z., & Zhu, Q. (2001). An approach for query optimizing in a mobile environment. *Proceedings of the Joint Conference on Information Systems (JCIS 2001)* (pp. 507-510).



## KEY TERMS

**Call-Back Locking (CBL):** CBL is an avoidance-based protocol that supports inter-transactional page caching. Transactions executing under an avoidance-based scheme must obey the read-once write-all (ROWA) replica management approach, which guarantees the correctness of data from the client cache by enforcing that all existing copies of an updated object have the same value when an updating transaction commits.

**Data Management for Mobile Computing:** Numerous database management issues exist in mobile computing environments, such as resource management and system support, representation/dissemination/management of information, location management, as well as others. Various new techniques for cache management, data replication, data broadcasting, transaction processing, failure recovery, as well as database security, have been developed. Applications of these techniques have been found distributed mobile database systems; mobile information systems; advanced mobile computing applications; and the Internet. Yet there are still many other issues need to be dealt with, such as the problem described in this article.

**Invalid Access Prevention Policy:** The invalid-access prevention policy requires that in order to manage multiple clients' concurrent read/write operations in the client/server architecture, no transactions that access stale multimedia data should be allowed to commit. In general, there are two different approaches to achieve this policy. The detection-based (lazy) policy ensures the validity of accessed multimedia data, and the avoidance-based (eager) policy ensures that invalid multimedia data is preemptively removed from the client caches.

**Multimedia Database:** A particular challenge for a multimedia database is the ability of dealing with multimedia data types. Retrieval of structured data from databases is typically handled by a database management system (DBMS), while retrieval of unstructured data from databases requires techniques developed for information retrieval (IR). (A survey on content-based retrieval for multimedia databases can be found in Yoshitaka and Ichikawa, A survey on content-based retrieval for multi-

media databases, *IEEE Transactions of Knowledge and Data Engineering*, 11(1), pp. 81-93, 1999.) Yet the rigid resource requirement demands more advanced techniques in dealing with multimedia objects in a mobile computing environment.

**Optimistic Two-Phase Locking (O2PL):** This is avoidance-based and is more optimistic about the existence of data contention in the network than CBL. It defers the write intention declaration until the end of a transaction's execution phase. Under the ROWA protocol, an interaction with the server is required only at client cache-miss or for committing its cache copy under the O2PL. As in CBL, all clients must inform the server when they erase a page from their buffer so that the server can update its page list.

**Pervasive Computing (or Ubiquitous Computing):** Pervasive computing "has as its goal the enhancing of computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user" (Mark Weiser, Hot topics: Ubiquitous computing, *IEEE Computer*, October 1993, p. 000). Pervasive computing is the trend towards increasingly ubiquitous, connected computing devices in the environment. As the result of a convergence of advanced electronic (particularly, mobile wireless) technologies and the Internet, pervasive computing is becoming a new trend of contemporary technology.

**Serializability:** Serializability requires that a schedule for executing concurrent transactions in a DBMS is equivalent to one that executes the transactions serially in a certain order.

**Server-Based Two-Phase Locking (S2PL):** The S2PL uses a detection-based algorithm and supports inter-transaction caching. It validates cached pages synchronously on a transaction's initial access to the page. Before a transaction is allowed to commit, it must first access the primary copies from the server on each data item that it has read at the client. The new value must be installed at the client if the client's cache version is outdated. The server is aware of a list of clients who requested locks only, and no broadcast is used by the server to communicate with clients.

# Enterprise Application Integration

**Christoph Bussler**

*Digital Enterprise Research Institute (DERI), Ireland*

## ENTERPRISE APPLICATION INTEGRATION (EAI) TECHNOLOGY

As long as businesses only have one enterprise application or back-end application system, there is no need to share data with any other system in the company. All data that has to be managed is contained within one back-end application system. However, as businesses grow, more back-end application systems find their way into their information technology infrastructure, managing different business data. These back-end application systems are not independent of each other; in general they contain similar business data or are part of the same business processes. This requires their integration for exchanging data between them. The technology that allows this is called enterprise application integration (EAI) technology. EAI technology is able to connect to back-end application systems in order to retrieve and to insert data. Once connected, EAI technology supports the definition of how extracted data is propagated to back-end application systems, solving the integration problem.

## BACKGROUND

Typical examples of back-end application systems that are deployed as part of a company's IT infrastructure are an enterprise resource planning (ERP) system and a manufacturing resource planning (MRP) system. In the general case, different back-end application systems store potentially different data about the same objects, like customers or machine parts. For example, a part might be described in an ERP as well as an MRP system. The reason for the part being described in two different back-end application systems is that different aspects of the same part are managed. In fact, this means that the not necessarily equal representation of the object exists twice, once in every system. If there are more than two systems, then it might be very well the case that the same object is represented several times. Any changes to the object have to be applied in every system that contains the object. And, since this cannot happen simultaneously in the general case, during the period of changing, the same object will be represented differently until the changes have been applied to all representations in all back-end application

systems. Furthermore, in most cases there is no record of how many systems represent the same object. It might very well be the case and actually often it is the case that a change is not applied to all objects because it is not known which back-end application system has a representation of the object in the first place.

In summary, the same object can be represented in different back-end application systems, the updates to an object can cause delays and inconsistencies, and locations of object representations can be unknown.

A second use case is that precisely the same object is replicated in different back-end application systems. In this case the update of the object in one system has to be applied to all the other systems that store the same object. The objects are replicas of each other since all have to be updated in the same way so their content is exactly the same. Only when all the objects are updated are they consistent again and the overall status across the back-end application systems is consistent again.

A third use case is that applications participate in common business processes. For example, first a part is being purchased through the ERP system, and upon delivery it is entered and marked as available in the MRP system. The business process behind this is consisting of several steps, namely, purchase a part, receive the part, make the part available, and so on. In this case the back-end application systems do not share common data, but their state depends on the progress of a business process, and it has to update the back-end application systems accordingly. In this sense they share a common business process, each managing the data involved in it.

All three use cases, while looking quite different from each other, have to be implemented by companies in order to keep their business data consistent. EAI technology (Bussler, 2003; Hohpe & Woolf, 2003) allows companies to accomplish this as it provides the necessary functionality as described next.

## Enterprise Application Integration Technology

Enterprise application integration technology addresses the various scenarios that have been introduced above by providing the required functionality. In the following, the different functionalities will be introduced step by step. First, EAI technology provides a reliable communication

mechanism so that any communication of data between back-end application systems is reliable. This ensures that no communication is lost. This is often achieved by sending messages through persistent and transactional queuing systems (Gray & Reuter, 1993). Queuing systems store messages persistently and transactionally, achieving an exactly-once message communication.

EAI technology uses adapters (J2EE Connector Architecture, 2004) in order to connect to the proprietary interfaces of back-end application systems. An adapter knows the proprietary interface and provides a structured interface to the EAI technology. This allows EAI technology to connect to any back-end application system for which an adapter is available. If no adapter is available for a particular back-end application system, then a new one has to be built. Some EAI technologies provide an adapter development environment allowing new adapters to be constructed. Adapters can retrieve data from back-end application systems as well as store data within back-end application systems. They therefore enable the communication of the EAI technology with the back-end application systems.

EAI technology by means of connecting to all required back-end application systems with adapters can propagate changes to all of the back-end application systems. This allows, for example, ensuring that all representations of an object in several back-end application systems can be changed as required. EAI technology is required to process changes in an ordered fashion to ensure object consistency. For example, an update of an object followed by a read of the object value must be executed in the given order. Any change in order would return an incorrect value (i.e., the value before instead of after the update). Any negative consequences of delay and inconsistencies are avoided by ensuring that every request is processed in the order of arrival.

In early implementations of EAI technology, publish/subscribe technology was used to establish the communication rules (Eugster, Felber, Guerraoui, & Kermarrec, 2003). A back-end system that has interest in specific objects and changes to them subscribes to those. Every back-end system publishes changes like object creation, update, or deletion. If there is a subscription that matches a publication, the same changes are applied to the subscribing back-end application system.

While publish/subscribe technology is quite useful, it is not able to implement business process integration (Bussler, 2003). In this case several back-end application systems have to be called in a specific order as defined by a business process. Sequencing, conditional branching, and other control flow constructs define the invocation order of the back-end application systems dynamically. Publish/subscribe cannot express this type of multistep execution, and more expressive technology is necessary.

Addressing business process-based integration, EAI technology incorporated workflow management systems in order to enable more complex and multistep executions. Through a workflow specification, the overall invocation order can be established, and complex processes like human resource hiring or supply chain management are possible. Workflow steps interact with back-end application systems through adapters, whereby the workflow steps themselves are sequenced by workflow definitions.

While business process-based integration is a significant step in the development of EAI technology, it is not sufficient in heterogeneous environments. For example, different back-end application systems can follow different data models. If this is the case, the same type of data is represented in different ways. For example, one back-end application system might implement an address as a record of street name, street number, city, and zip code while another back-end application system might implement an address as two strings, address line 1 and address line 2. If data is transmitted from one back-end application system to another one, the address has to be mediated from one representation to the other representation for the data to be understood by the receiving back-end application system.

## **CURRENT DEVELOPMENTS AND FUTURE TRENDS**

The latest development in EAI technology takes data heterogeneity into consideration and provides mediation technology. For example, Microsoft's Biztalk Server (Microsoft, 2004) has a mapping tool that allows the graphical definition of mapping rules that can transform a message in one format into a message of a different format. Tools from other vendors follow the same approach. The research community is also working on the data mediation problem and an overview is given in Rahm and Bernstein (2001).

Also, this new generation realizes that back-end application systems expose not only data but also public processes that define interaction sequences of data (Bussler, 2003). A public process allows one to determine which messages a back-end application systems sends or receives and in which order. This allows ensuring that all messages are exchanged appropriately with the back-end application system.

Web services are a relatively new approach to communicate data and messages over the Internet (Alonso, Casati, Kuno, & Machiraju, 2004). Web services are starting to be used in EAI technologies in several places. One is for the communication between back-end application systems and the EAI technology. The other place is to define the interface of back-end application systems as

Table 1. A summary of critical issues

<p><b>Autonomy</b> Back-end applications are autonomous in their state changes, and EAI technology must be aware that it cannot control the application's state changes.</p> <p><b>Communication</b> EAI technology must provide reliable and secure communication between back-end applications in order to insure overall consistency.</p> <p><b>Delay</b> Changes of objects represented in several back-end application systems might not happen instantaneously due to the time required to do the update, and so a delay can occur between the update of the first and the last change.</p> <p><b>Distribution</b> Back-end application systems are distributed in the sense that each has its own separate storage or database management systems, with each separately controlled and managed.</p> <p><b>Duplication</b> Objects that have to reside in several back-end application systems might require duplication in order to ensure back-end application state consistency.</p> <p><b>Heterogeneity</b> Back-end application systems are implemented based on their own data model, and due to the particular management focus, the data models differ in general, not complying with a common standard.</p> <p><b>Inconsistency</b> If a delay happens while changing different representations of the same object,</p>	<p>inconsistencies can occur if these different representations are accessed concurrently.</p> <p><b>Mediation</b> Due to heterogeneity of back-end applications, objects are represented in different data models that have to be mediated if objects are sent between applications.</p> <p><b>Process management</b> Multistep business processes across back-end application systems require process management to implement the particular invocation sequence logic.</p> <p><b>Publish/subscribe</b> Back-end application systems can declare interest in object changes through subscriptions that are matched with publications of available object changes.</p> <p><b>Reliability</b> The integration of back-end application systems must be reliable in order to neither lose data nor accidentally introduce data by retry logic in case of failures.</p> <p><b>Replication</b> Replication is a mechanism that ensures that changes of an object are automatically propagated to duplicates of this object. The various representations act as if they are one single representation.</p> <p><b>Security</b> Communication of data between back-end application systems through EAI technology must ensure security to avoid improper access.</p>
--	---

Web services. In this case all back-end application systems have a homogeneous interface technology they expose, and an EAI technology does not have to deal with the huge variety of interfaces of back-end application systems anymore.

Further out are Semantic Web technology-based approaches. They are in a research state today but can be expected to be available in commercial technology later on. A significant effort applying Semantic Web technology in order to solve the integration problem is the Web Service Modeling Ontology (WSMO, 2004). This ontology, which was developed by a significant number of organizations, uses Semantic Web technology to define a conceptual model to solve the integration problem that encompasses EAI integration. The main conceptual elements proposed are ontologies for defining data and messages, mediators for defining transformation, goals for dynamically binding providers, and Web services for defining the interfaces of communicating services. In order to define a specific EAI integration, the formal Web Service Modeling Language (WSML, 2004) is defined. This can be processed by a

WSML parser. In order to execute B2B integration, the Web Service Modeling Ontology Execution environment (WSMX, 2004) was developed. It serves as a runtime environment for integrations defined through WSML.

## CRITICAL ISSUES OF EAI TECHNOLOGIES

The critical issues of enterprise application integration are listed in Table 1. Current generations of EAI technology have to address these issues in order to be competitive and considered appropriate implementations to the EAI problem. The same applies to research. Recently, research as well as new products focuses on Web services as a means to integrate applications. However, Web services are only a new technology, and all the requirements discussed above and issues listed below still apply. Semantic Web services (WSMO, 2004) addresses the requirements and issues, taking all aspects into consideration.

## CONCLUSION

Enterprise application integration (EAI) technology is essential for enterprises with more than one back-end application system. Current EAI technology is fairly expressive, being able to handle most of the integration tasks. Newer developments like Web services (2004) and Semantic Web services (WSMO, 2004) will significantly improve the situation by introducing semantic descriptions, making integration more reliable and dependable.

## REFERENCES

- Alonso, G., Casati, F., Kuno, H., & Machiraju, V. (2004). *Web services: Concepts, architectures and applications*. Berlin, Germany: Springer-Verlag.
- Bussler, C. (2003). *B2B integration*. Berlin, Germany: Springer-Verlag.
- Eugster, P., Felber, P., Guerraoui, R., & Kermarrec, A.-M. (2003). The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35(2).
- Gray, J., & Reuter, A. (1993). *Transaction processing: Concepts and techniques*. Morgan Kaufmann.
- Hohpe, G., & Woolf, B. (2003). *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Boston: Addison-Wesley.
- J2EE Connector Architecture (2004). *J2EE Connector Architecture*. [java.sun.com/j2ee/connector/](http://java.sun.com/j2ee/connector/)
- Microsoft (2004). Microsoft Corporation. [www.microsoft.com](http://www.microsoft.com)
- Rahm, E., & Bernstein, P. (2001). A survey of approaches to automatic schema matching. *VLDB Journal*, 10, 334-350.
- Web Services (2004). Web Service Activity. [www.w3.org/2002/ws/](http://www.w3.org/2002/ws/)
- WSML (2004). Web Service Modeling Language. [www.wsmo.org/wsml](http://www.wsmo.org/wsml)
- WSMO (2004). Web Service Modeling Ontology. [www.wsmo.org](http://www.wsmo.org)
- WSMX (2004). Web Service Modeling Execution Environment. [www.wsmx.org](http://www.wsmx.org)

## KEY TERMS

**Adapters:** Adapters are intermediate software that understand proprietary back-end application interfaces and provide easy access interfaces for EAI technology integration.

**Back-End Application Systems:** Software systems managing business data of corporations.

**EAI Technology:** Software systems that provide business-to-business integration functionality by sending and receiving messages and retrieving and storing them in back-end application systems.

**Process Management:** Process management allows defining specific invocation sequences of back-end application systems in the context of EAI.

**Publish/Subscribe:** Publish/subscribe is a technology whereby interest can be declared and is matched upon the publication of object changes.

**Workflow Technology:** Workflow technology is a software component that provides the languages and interpreters to implement process management.



# Extended Entity Relationship Modeling

Sikha Bagui

University of West Florida, USA

## INTRODUCTION

With the rising complexity of database applications, the basic concepts of Entity-Relationship (ER) modeling (as originally developed by Chen, 1976) were no longer sufficient. So the basic ER model was extended to include generalizations and specializations (Bagui & Earp, 2003; Elmasri & Navathe, 2003) and the concept of categories (Elmasri, Weeldreyer & Hevner, 1985). An ER model which includes all the concepts of the original ER model and the additional concepts of generalizations/specializations and categories is often referred to as the Extended ER (EER) model (Elmasri & Navathe, 2003). In this short paper, we shed some light on these relationship concepts, concepts that database designers often find difficult to model directly (Engels et al., 1992/1993). We also discuss the mapping rules for generalizations/specializations and categories. Important contributions in this area are also reported in (Elmasri et al., 1985; Markowitz & Shoshani, 1992; Dey, Storey & Barron, 1999; Wiederhold & Elmasri, 1980). Song, Evans, and Park (1995) also give a good comparative analysis of other types of ER diagrams and their notations.

Due to the short nature of this paper, we will keep the discussion focused on implementing generalizations and specializations in relational databases; their parallel implementation in objects will not be covered. Also, the discussion of the concept of inheritance will center around generalizations/specializations and categories in EER diagrams, without getting into an almost equivalent notion in object-oriented (OO) theory, ORM (Object-Role Modeling), and UML (Unified Modeling Language) class diagrams.

## BACKGROUND

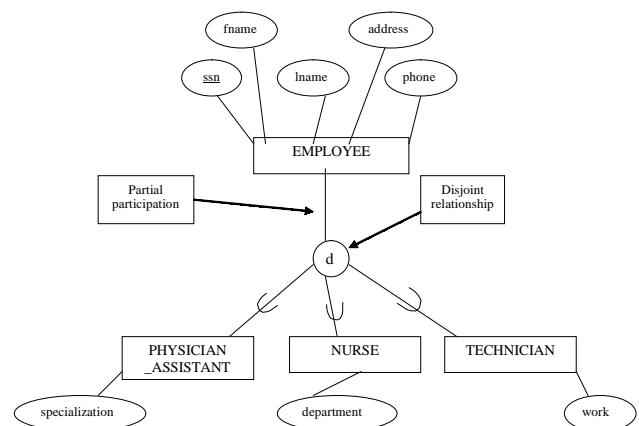
A generalization/specialization relationship models a superclass/subclass type of relationship. A generalization is an abstraction mechanism that allows for viewing of entity-sets as a single generic entity-set. The attributes and associations which are common to the entity-sets are associated with the generic (generalized) entity-set. The inverse of generalization is called specialization.

## GENERALIZATION/ RELATIONSHIPS

If we are modeling a hospital database, and we want to store information about the hospital's nurses, technicians, and physician assistants, we could create separate entities such as NURSE, TECHNICIAN, and PHYSICIAN ASSISTANT. But, these three entities would also have a lot of fields in common, for example, name, social security number, address, phone, and so forth. So, it would be a good idea to have an entity set called EMPLOYEE containing these common fields, and entity subsets, NURSE, TECHNICIAN and PHYSICIAN ASSISTANT, that could inherit this information from the EMPLOYEE entity set. In this case, the EMPLOYEE entity set would be called the *superclass*. This superclass is a *generalization* of the entity subsets, NURSE, TECHNICIAN, and PHYSICIAN ASSISTANT. The NURSE, TECHNICIAN, and PHYSICIAN ASSISTANT would be called the *subclasses*. The subclasses are *specializations* of the superclass, EMPLOYEE, and inherit from the superclass. Several specializations can be defined for the same entity type (or superclass).

The subclass, denoted by a separate entity rectangle in the EER diagram, is considered to be a *part* of the superclass entity set, EMPLOYEE. Although it will have attributes that distinguish it from other subclasses, it is considered only a subset of the EMPLOYEE entity set. That is, all nurses are employees, but the reverse is not true—not all employees are nurses. Likewise, all techni-

Figure 1. A generalization/specialization relationship



cians or physician assistants are employees, but all employees are not technicians or physician assistants.

Figure 1 shows this generalization/specialization example. We use Elmasri and Navathe's (2003) diagrammatic notations for the EER diagrams. The subset symbol, " $\subset$ ", indicates the direction of the superclass/subclass or parent-child, inheritance relationship. This superclass/subclass relationship is also often referred to as a *IS-A* or *IS-PART-OF* relationship (Sanders, 1995).

## Constraints on Generalization/Specialization Relationships

Generalizations and specializations can have two types of constraints: (1) the *disjoint/overlap* relationship constraint and (2) participation constraints – total or partial. The combinations of these constraints can be (1) disjoint and total participation; (2) disjoint and partial participation; (3) overlap and total participation; or (4) overlap and partial participation. First we will discuss disjoint/overlap relationship constraints, and then we will discuss participation constraints, giving examples of combinations of the constraints along the way.

### Disjoint/Overlap Relationship Constraints

Generalization/specialization relationships may be *disjoint* or they may *overlap*. A disjoint relationship is shown by a "d" in the circle attaching the superclass to the subclass or subclasses (as shown in Figure 1). A disjoint relationship means that an entity from the superclass can belong to only one of the subclasses (can be of only one specialization). For example, according to Figure 1, an EMPLOYEE can be at most a member of only one of the subclasses – PHYSICIAN ASSISTANT, NURSE, or TECHNICIAN. An employee cannot be a physician assistant as well as a nurse, or cannot be a nurse as well as a technician.

An overlap relationship is shown by an "o" in the circle attaching the superclass to the subclass or subclasses (as shown in Figure 4). Overlap means that an entity from the superclass can belong to more than one subclass (specialization). For example, according to Figure 4, a computer must be either a laptop or a desktop, or both a laptop and a desktop.

### Participation Constraints

The second type of constraint on generalization/specialization relationships is participation constraints, which may be *total* (or full) or *partial*. As in the ER model (Bagui

& Earp, 2003; Elmasri & Navathe, 2003), we will show a full or total participation between the superclass and subclass entities by double lines, and a partial participation between the superclass and subclass entities by single lines. Partial participation is shown in Figure 1. Figure 1 can be read as:

*An EMPLOYEE may either be a PHYSICIAN ASSISTANT, NURSE, or TECHNICIAN.*

Figure 1 shows a *disjoint, partial participation* relationship. The "may" means partial participation between the EMPLOYEE superclass and the respective subclasses entities. That is, not all employees of the EMPLOYEE entity set belong one of the subclasses, PHYSICIAN ASSISTANT, NURSE, or TECHNICIAN. One may ask, why? Or how? To answer this, we will extend Figure 1 to include another subclass, as shown in Figure 2. We now have an Employee from the EMPLOYEE entity set who may also belong to the HEAD subclass. Once again, the "may" is inferred from the single line between the superclass (or generalization) entity, EMPLOYEE, and the subclass (or specialization) entity, HEAD. Figure 2 can be read as:

*An Employee may be a HEAD or PHYSICIAN ASSISTANT or NURSE or TECHNICIAN. Or, an Employee may be both a HEAD and a PHYSICIAN ASSISTANT, or both a HEAD and a NURSE, or both a HEAD and a TECHNICIAN.*

An example of total or full participation is shown in Figure 3. We can read Figure 3 as:

*An EMPLOYEE must either be a PHYSICIAN ASSISTANT, NURSE, or TECHNICIAN.*

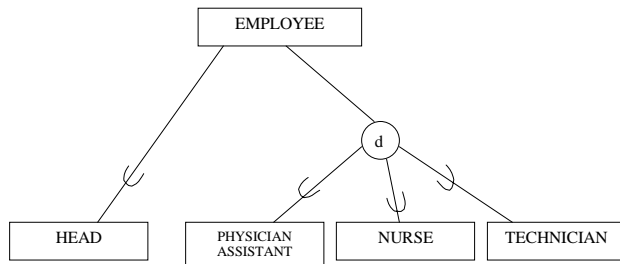
The "must" means total participation. So, an EMPLOYEE must belong to one of the subclasses. That is, *all* employees of the EMPLOYEE entity set must belong to one of the subclasses. But although there is total participation in Figure 3, the employee cannot belong to more than one subclass because of the "d" or disjoint relationship. Figure 3 shows a *disjoint, full participation* relationship, and Figure 4 shows an *overlap, full participation* relationship.

## Mapping Generalizations and Specializations to a Relational Database

Rules to map generalizations/specializations to a relational database depend on the constraints on the generali-

## Extended Entity Relationship Modeling

Figure 2. A disjoint



zation/specialization relationships. One of the following four rules are generally used (Elmsari & Navathe, 2003) to map generalizations and specializations to a relational database:

### Rule 1

Rule 1 works well for total or partial generalization/specialization relationships as well as disjoint or overlap generalization/specialization relationships. Using this rule, we would create a separate relation for the superclass as well as for each of the subclasses.

For rule 1:

1. Create a relation for the superclass entity. Include all the attributes of this entity in this relation and underline the primary key attribute.
2. Create a separate relation for each subclass (specialization) entity. Include the attributes of the respective subclasses in the respective subclass relations. Include the primary key from the superclass entity or relation in the subclass relation as the primary key of the subclass relation (and underline it).

Figure 3. A disjoint and full participation with predicate defined subclasses

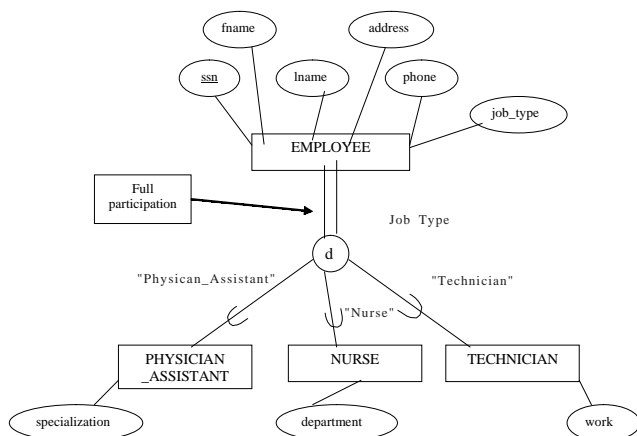
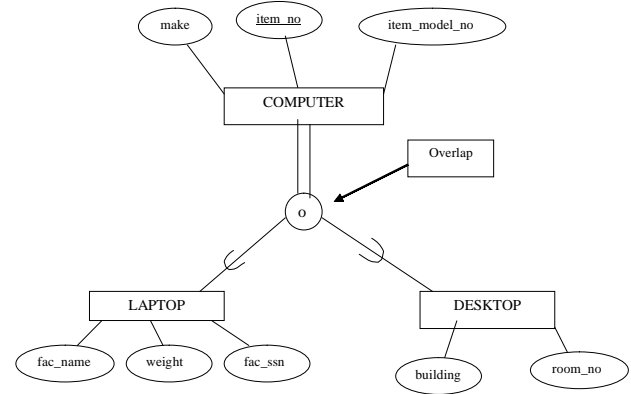


Figure 4. An overlap and full participation



To illustrate this rule, we will map Figure 1 as shown in the following example:

EMPLOYEE				
<u>ssn</u>	fname	lname	address	phone

PHYSICIAN ASSISTANT	
<u>ssn</u>	specialization

NURSE	
<u>ssn</u>	department

TECHNICIAN	
<u>ssn</u>	work

### Rule 2

Rule 2 works well if: (a) there is total or full participation between the superclass and the subclasses. That is, if every member of the superclass entity set belongs to at least one of the subclasses; (b) if there is a disjoint relationship—otherwise there will be redundancy if a superclass entity belongs to more than one subclass; and (c) when there is more emphasis on the subclass (specialization), and it is more important to know about the subclass and its attributes. By this rule, we create a separate relation for each subclass. In this rule, you do not have a separate relation for the superclass entity.

For rule 2: Create a separate relation corresponding to each subclass entity. Include the attributes of the respective subclasses in the respective subclass relations. Also include the primary key and other attributes of the superclass entity in all the subclass relations. Underline the primary key brought from the superclass entity (to the subclass relation).

To illustrate this rule, we will map Figure 1 (but we are assuming that Figure 1 has full participation—double lines—between the EMPLOYEE superclass and the subclasses):

PHYSICIAN ASSISTANT					
ssn	specialization	fname	lname	address	phone

NURSE					
ssn	department	fname	lname	address	phone

TECHNICIAN					
ssn	work	fname	lname	address	phone

### Rule 3

Rule 3 works well if: (a) there is a disjoint relationship between the superclass and subclasses. It will create redundancy in the database if used with overlap scenarios. And, (b) if the subclasses are predicate defined (condition defined) or attribute defined. A predicate defined subclass is where a condition is placed on the value of some attribute of the superclass to determine the subclass. Figure 3 shows an example of a predicate defined subclass. If, as shown in Figure 3, the EMPLOYEE entity has an additional attribute, JobType, and a condition is specified on the condition of membership in the PHYSICIAN ASSISTANT subclass by the condition (jobType="Physician\_Assistant"), this is a defining predicate of the subclass, PHYSICIAN ASSISTANT. A predicate-defined subclass is shown by writing the predicate condition next to the arc that connects the subclass to the circle. Also, the defining attribute name is placed on the arc from the superclass to the circle. This rule is not recommended when subclasses have too many attributes (since this will cause too many null values).

For rule 3: Create a single relation that includes the superclass and its attributes as well as the subclasses and its attributes. For this rule, we will map Figure 3 as shown below:

EMPLOYEE								
ssn	fname	lname	address	phone	jobtype	specialization	department	work

### Rule 4

Rule 4 works for both overlaps and disjoints, but it works better for overlaps. With disjoints, this rule will create null values when the entity is not a member of a particular subclass. This rule is also not recommended when subclasses have too many attributes (since this will also cause too many null values). If subclasses have few attributes, however, this rule may be preferred to rule 1 or rule 2 since it will eliminate the need for a relational join.

In this rule, a flag is created for each superclass tuple that belongs to a subclass.

For rule 4: Create a single relation that includes the attributes of the superclass and the attributes of its subclasses. To illustrate this rule, we will map Figure 4 as shown below:

COMPUTER									
item_no	make	item_model_no	lflag	fac_name	weight	fac_ssn	dflag	building	room_no

## CATEGORIES

The concept of categories extends the concepts of generalization entity types and specialization entity types even further. Categories are created by grouping entity types (generalization entity types or specialization entity types) by the roles they may play within relationships. So, categories can represent superclasses (generalization categories) or subclasses (specialization categories).

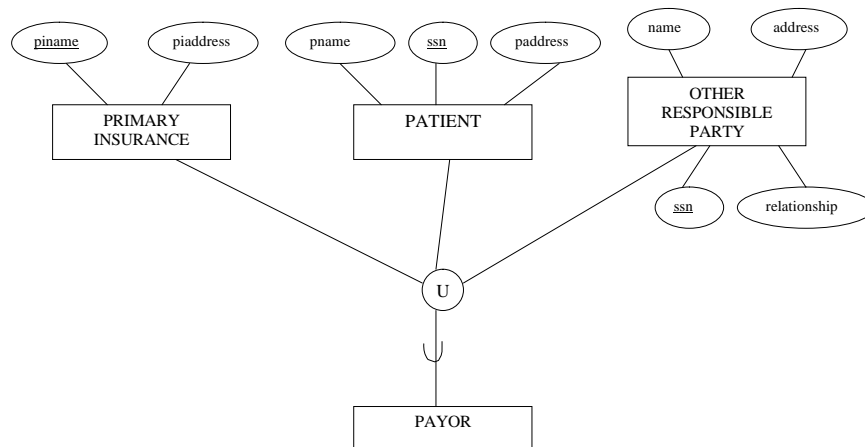
In Figure 5, the PAYOR could be inheriting from the PRIMARY INSURANCE, PATIENT, or OTHER RESPONSIBLE PARTY. The PRIMARY INSURANCE, PATIENT, and OTHER RESPONSIBLE PARTY represent a superclass (generalization category) of entity types. Each of the entity types in this generalization or superclass is a different entity type with different keys, but they play a common role – the role of a PAYOR. Here, we would refer to the subclass (in this case, PAYOR) as a *category* or *union type* of two or more superclasses. Hence, a category is a subclass of a union of two or more superclasses that are *different* entity types (Elmasri et al., 1985; Elmasri & Navathe, 2003) playing a common role. A category is diagrammatically shown by a “∪” in the circle that attaches the category to the superclasses, as shown in Figure 5. We can read Figure 5 as:

*A PAYOR may be a PRIMARY INSURANCE or PATIENT or OTHER RESPONSIBLE PARTY.*

## Participation Constraints in Categories

Just like other subclasses, categories can also have total (or full) participation or partial participation. Total participation means that the category holds the union of *all* entities in its superclasses. That is, if there was full participation in Figure 5 (double lines running from the category, PAYOR, to the circle with the “∪”), then every entity of PRIMARY INSURANCE would exist in PAYOR, every entity of PATIENT would exist in PAYOR, and every entity of OTHER RESPONSIBLE PARTY would exist in PAYOR.

Figure 5. A category



Partial participation means that a category holds only a subset of the union of all entities in its superclasses. That is, as shown in Figure 5, every entity of PRIMARY INSURANCE does not exist in PAYOR, every entity of PATIENT does not exist in PAYOR, and every entity of OTHER RESPONSIBLE PARTY does not exist in PAYOR. Once again, partial participation would be shown by single lines from the category to the circle with the “ $\cup$ ”.

### Mapping Categories

There are two rules to map categories (Elmasri & Navathe, 2003).

#### Rule 5

Rule 5 should be used when the superclasses have different keys. For rule 5:

1. Create a new relation to correspond to the category.
2. Since the keys of each of the superclasses are different, we cannot use any one of them as the key for this new relation. So, we have to specify a new key attribute (called a surrogate key) to correspond to the category in the new relation.
3. To be able to join the subclass with the superclass/superclasses, include the surrogate key attribute as the foreign key in each superclass relation.

To illustrate rule 5, we will map Figure 5 as shown below:

PRIMARY INSURANCE				
<u>piname</u>	piaddress	payorid		

PATIENT				
<u>ssn</u>	paddress	pname	payorid	

OTHER RESPONSIBILITY PARTY				
<u>ssn</u>	relationship	name	address	payorid

PAYOR	
<u>payorid</u>	

#### Rule 6

The second rule used to map categories is used when the superclass entities have the same key.

For example, we would map Figure 6 as:

DORM	
<u>dormnu</u>	dname

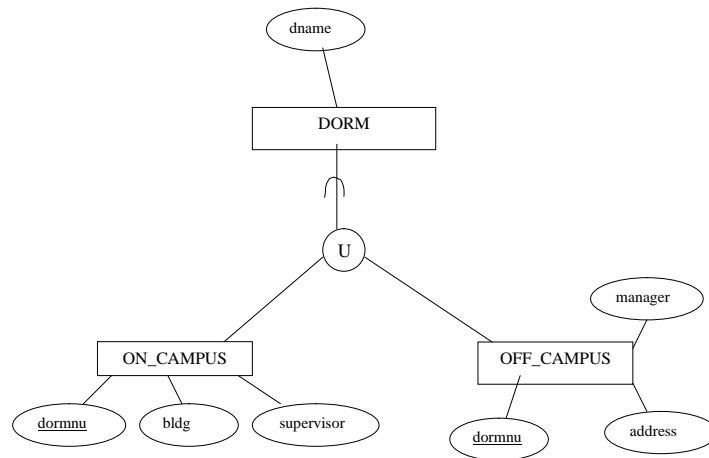
ON CAMPUS		
<u>dormnu</u>	bldg	supervisor

OFF CAMPUS		
<u>dormnu</u>	address	manager



Figure 6. Category where superclasses have the same key



## Multiple Inheritance

In this article, we have given examples of how subclasses inherit from superclasses. In reality, however, subclasses often inherit from more than one superclass. This concept is known as multiple inheritance. Categories are also not necessarily disjoint, so a given entity may be a member of several different categories. Due to the brief nature of this paper, we will not elaborate on the concept of multiple inheritance.

## FUTURE TRENDS

Due to the dominance of relational DBMSs and the intuitive nature of the ER approach, EER modeling is still being widely used. But, with the increasing trend towards object-oriented programming, the database world is likely to see an increase in the use of other models like ORM and UML in the future. ORMs view the world in terms of objects playing roles. ORMs are more expressive than ER models in capturing constraints or business rules and offer greater semantic stability than ER models (Halpin, 2001). Further advantages and disadvantages of ORM modeling are reported in Halpin (2000a, 2000b, 2001).

To date, UML is mainly used in industry for designing object-oriented program code (Halpin, 2001). Though sometimes used for designing databases, UML has so far had little success in displacing approaches such as ER and EER. Nevertheless, UML could very well become popular for database design in the future (Halpin, 2001). Further advantages and disadvantages of UML modeling versus ER modeling are reported in Halpin and Bloesch (1999); Hay (1999); Kobryn (1999); and Muller (1999).

## CONCLUSION

Databases are becoming increasingly complex today. To deal with these complexities, it is becoming essential for database designers to have an understanding of the extended ER model, which incorporates generalizations/specializations and categories. In this paper, we briefly presented generalizations/specializations and categories with the use of examples. We also presented rules to map generalizations/specializations and categories to a relational database.

## REFERENCES

- Bagui, S., & Earp, R. (2003). *Database design using entity-relationship diagrams*. Boca Raton, FL: CRC Press/Auerbach.
- Chen, P.P. (1976). The entity-relationship model: Toward a unified view of data. *ACM Transactions of Database Systems*, 1(1), 9-36.
- Dey, D., Storey, V., & Barron, T. (1999). Improving database design through the analysis of relationships. *ACM Transactions on Database Systems*, 24(4), 453-486.
- Elmasri, R., & Navathe, S.B. (2003). *Fundamentals of database systems* (4th ed.). Addison-Wesley.
- Elmasri, R., Weeldreyer, J., & Hevner, A. (1985). The category concept: An extension to the entity-relationship model. *Data and Knowledge Engineering*, 1, 75-116.
- Engels, G., Gogolla, M., Hohenstein, U., Hulsmann, K., Lohr-Richter, P., Saake, G., & Ehrlich, H.-D. (1992/93).

Conceptual modeling of database applications using an extended ER model. *Data and Knowledge Engineering*, 9, 157-204.

Halpin, T.A. (2000a). An ORM metamodel. *Journal of Conceptual Modeling*, 9.

Halpin, T.A. (2000b). An ORM metamodel of Barker ER. *Journal of Conceptual Modeling*, 9.

Halpin, T.A. (2001). *Information modeling and relational databases*. Morgan Kaufmann.

Halpin, T.A., & Bloesch, A.C. (1999). Data modeling in UML and ORM: A comparison. *Journal of Database Management*, 10(4), 4-13.

Hay, D.C. (1999). There is no object-oriented analysis. *DataToKnowledge Newsletter*, 27(1).

Kobryn, C. (1999). UML 2001: A standardization odyssey. *Communication of the ACM*, 42(10), 29-37.

Markowitz, V., & Shoshani, A (1992). Representing extended entity-relationship structures in relational databases: A modular approach. *ACM Transactions on Database Systems*, 17(3), 423-464.

Muller, R.J. (1999). *Database design for smarties*. San Francisco: Morgan Kaufmann.

Sanders, L.G. (1995). *Data modeling*. Thomson International.

Song, I.Y., Evans, M., & Park, E.K. (1995). A comparative analysis of entity-relationship diagram. *Journal of Computer and Software Engineering*, 3(4), 427-459.

Wiederhold, G., & Elmasri, R. (1980). The structural model for database design. In P.P. Chen (Ed.), *Entity-relationship approach to systems analysis and design*. Amsterdam: North-Holland.

## KEY TERMS

**Category:** A collection of objects or entities that is a subset of the union of different entity types; entities offering a similar role are grouped into a category.

**Entity Sets or Entity Types:** Similar entities or objects with common properties are summarized into entity sets or entity types; graphically represented in rectangles.

**Generalization:** When entities have similar basic attributes, they can be classified into a generalized entity type. Generalization is the process of finding commonalities between entities to be able to abstract to a higher level entity set.

**Inheritance:** A subclass inherits all the attributes of a superclass and all the relationships that it (the superclass) participates in.

**Specialization:** A process of conceptual refinement to form specialized subclasses for entity sets. An entity type can have several specializations or categories of specializations defined on the basis of some distinguishing characteristics of the entities in the superclass.

**Subclass:** Same as specialization; a meaningful subgrouping of entity sets that needs to be represented explicitly. These subgroups are a subset of the entities that belong to the entity set from which they are derived.

**Superclass:** Same as generalization. A superclass is the main set (entity) from which subsets (subclasses) are defined based on meaningful criteria needed for a particular database.

# Extraction–Transformation–Loading Processes

**Alkis Simitsis**

*National Technical University of Athens, Greece*

**Panos Vassiliadis**

*University of Ioannina, Greece*

**Timos Sellis**

*National Technical University of Athens, Greece*

## INTRODUCTION

A data warehouse (DW) is a collection of technologies aimed at enabling the knowledge worker (executive, manager, analyst, etc.) to make better and faster decisions. The architecture of a DW exhibits various layers of data in which data from one layer are derived from data of the lower layer (see Figure 1). The operational databases, also called data sources, form the starting layer. They may consist of structured data stored in open database and legacy systems, or even in files. The central layer of the architecture is the global DW. The global DW keeps a historical record of data that result from the transformation, integration, and aggregation of detailed data found in the data sources. An auxiliary area of volatile data, data staging area (DSA) is employed for the purpose of data transformation, reconciliation, and cleaning. The next layer of data involves client warehouses, which contain highly aggregated data, directly derived from the global warehouse. There are various kinds of local warehouses, such as data mart or on-line analytical processing (OLAP) databases, which may use relational database systems or specific multidimensional data structures. The whole environment is described in terms of its components, metadata, and processes in a central metadata repository, located at the DW site.

In order to facilitate and manage the DW operational processes, specialized tools are available in the market,

under the general title extraction-transformation-loading (ETL) tools. ETL tools are pieces of software responsible for the extraction of data from several sources, their cleansing, customization, and insertion into a DW (see Figure 2). The functionality of these tools includes

- the *extraction* of relevant information at the source side;
- the *transportation* of this information to the DSA;
- the *transformation* (i.e., customization and integration) of the information coming from multiple sources into a common format;
- the *cleaning* of the resulting data set, on the basis of database and business rules; and
- the *propagation and loading* of the data to the DW and the *refreshment* of data marts.

## BACKGROUND

In the past, there have been research efforts towards the design and optimization of ETL tasks. We mention three research prototypes: (a) AJAX, (b) Potter's Wheel, and (c) ARKTOS II. The first two prototypes are based on algebras, which we find mostly tailored for the case of homogenizing Web data; the latter concerns the modeling of ETL processes in a customizable and extensible manner.

Figure 1. A data warehouse environment

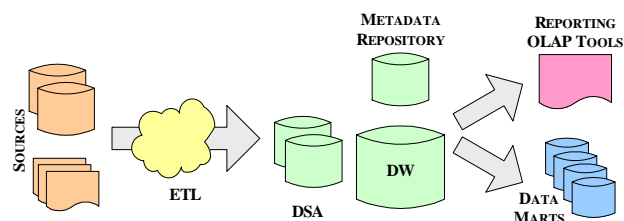
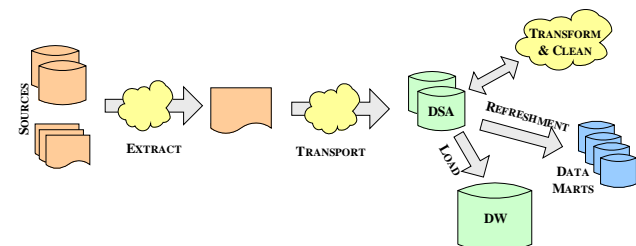


Figure 2. ETL processes in detail



The AJAX system (Galhardas, Florescu, Sasha, & Simon, 2000) deals with typical data quality problems, such as the object identity problem, errors due to mistyping, and data inconsistencies between matching records. This tool can be used either for a single source or for integrating multiple data sources. AJAX provides a framework wherein the logic of a data cleaning program is modeled as a directed graph of data transformations that starts from some input source data. AJAX also provides a declarative language for specifying data cleaning programs, which consists of SQL statements enriched with a set of specific primitives to express mapping, matching, clustering, and merging transformations. Finally, an interactive environment is supplied to the user to resolve errors and inconsistencies that cannot be automatically handled and to support a stepwise refinement design of data cleaning programs.

Raman and Hellerstein (2001) present the Potter’s Wheel system, which is targeted to provide interactive data cleaning to its users. The system offers the possibility of performing several algebraic operations over an underlying data set, including format (application of a function), drop, copy, add a column, merge delimited columns, split a column on the basis of a regular expression or a position in a string, divide a column on the basis of a predicate (resulting in two columns, the first involving the rows satisfying the condition of the predicate and the second involving the rest), selection of rows on the basis of a condition, folding columns (where a set of attributes of a record is split into several rows), and unfolding. Optimization algorithms are also provided for the CPU usage for certain classes of operators. The general idea behind Potter’s Wheel is that users build data transformations in an iterative and interactive way; thereby, users can gradually build transformations as discrepancies are found and clean the data without writing complex programs or enduring long delays.

ARKTOS II is a coherent framework for the conceptual, logical, and physical design of ETL processes. The goal of this line of research is to facilitate, manage, and optimize the design and implementation of the ETL processes, during both the initial design and deployment stage, as

such during the continuous evolution of the DW. To this end, Vassiliadis, Simitis, and Skiadopoulos (2002) and Simitis and Vassiliadis (2003) proposed a novel conceptual model. Further, Simitis, Vassiliadis, Skiadopoulos, and Sellis (2003) and Vassiliadis et al. (2004) presented a novel logical model. The proposed models, conceptual and logical, were constructed in a customizable and extensible manner so that the designer can enrich them with his own reoccurring patterns for ETL processes. Therefore, ARKTOS II offers a palette of several templates, representing frequently used ETL transformations along with their semantics and their interconnection (see Figure 3). In this way, the construction of ETL scenarios as a flow of these transformations, is facilitated. Additionally, ARKTOS II takes into account the optimization of ETL scenarios, with a main focus on the improvement of the time performance of an ETL process, and ARKTOS II tackles the problem of how the software design of an ETL scenario can be improved, without any impact on its consistency.

An extensive review of data quality problems and related literature, along with quality management methodologies can be found in Jarke, Lenzerini, Vassiliou, and Vassiliadis (2000). Rundensteiner (1999) offered a discussion on various aspects of data transformations. Sarawagi (2000) offered a similar collection of papers in the field of data, including a survey (Rahm & Do, 2000) that provides an extensive overview of the field, along with research issues and a review of some commercial tools and solutions for specific problems (e.g., Borkar, Deshmuk, & Sarawagi, 2000; Monge, 2000). In a related but different context, the IBIS tool (Calì et al., 2003) is an integration tool following the global-as-view approach to answer queries in a mediated system. Moreover, there exists a variety of ETL tools in the market. Simitis (2004) listed the ETL tools available at the time this paper was written.

**MAIN THRUST OF THE ARTICLE**

In this section we briefly review the individual issues that arise in each of the phases of an ETL process, as well as the problems and constraints that concern it.

*Figure 3. Typical template transformations provided by ARKTOS II*

<b>Filters</b>	<b>Unary transformations</b>	<b>Binary transformations</b>
Selection ( $\sigma$ )	Push	Union (U)
Not null (NN)	Aggregation ( $\gamma$ )	Join ( $\bowtie$ )
Primary key violation (PK)	Projection ( $\pi$ )	Diff ( $\Delta$ )
Foreign key violation (FK)	Function application (f)	Update Detection ( $\Delta_{UPD}$ )
Unique value (UN)	Surrogate key assignment (SK)	<b>Composite transformations</b>
Domain mismatch (DM)	Tuple normalization (N)	Slowly changing dimension (Type 1,2,3)(SDC-1/2/3)
<b>Transfer operations</b>	Tuple denormalization (DN)	Format mismatch (FM)
Ftp (FTP)	<b>File operations</b>	Data type conversion (DTC)
Compress/Decompress (Z/dZ)	EBCDIC to ASCII conversion (EB2AS)	Switch ( $\sigma^*$ )
Encrypt/Decrypt (Cr/dCr)	Sort file (Sort)	Extended union (U)

## Global Problems and Constraints

Scalzo (2003) mentions that 90% of the problems in DW arise from the nightly batch cycles that load the data. At this period, administrators have to deal with problems such as (a) efficient data loading, and (b) concurrent job mixture and dependencies. Moreover, ETL processes have global time constraints, including the time they must be initiated and their completion deadlines. In fact, in most cases, there is a tight time window during the night that can be exploited for the refreshment of the DW, because the source system is off-line or not heavily used during this period.

Consequently, a major problem arises with the scheduling of the overall process. The administrator must find the right execution order for dependent jobs and job sets on the existing hardware for the permitted time schedule. On the other hand, if the OLTP applications cannot produce the necessary source data in time for processing before the DW comes online, the information in the DW will be out of date. Still, because DWs are used for strategic purposes, this problem can be afforded because long-term reporting and planning is not severely affected by this type of failure.

## Extraction and Transportation

During the ETL process, one of the first tasks that must be performed is the extraction of relevant information that must be further propagated to the warehouse. To minimize the overall processing time, this involves only a fraction of the source data that has changed since the previous execution of the ETL process and mainly concerns the newly inserted, and possibly updated, records. Usually, change detection is physically performed by the comparison of two snapshots (one corresponding to the previous extraction and the other corresponding to the current one). Efficient algorithms exist for this task, such as the snapshot differential algorithms presented by Labio and Garcia-Molina (1996). Another technique is log sniffing, (i.e., the scanning of the log file to reconstruct the changes performed since the last scan). In rare cases, change detection can be facilitated by the use of triggers. However, this solution is technically impossible for many of the sources that consist of legacy systems or plain flat files. In numerous other cases, where relational systems are used at the source side, the usage of triggers is also prohibitive due to the performance degradation that their usage incurs and the need to intervene in the structure of the database. Moreover, another crucial issue concerns the transportation of data after the extraction, where tasks such as FTP, encryption–decryption, compression–decompression, and so forth, can take place.

## Transformation and Cleaning

It is possible to determine typical tasks that take place during the transformation and cleaning phase of an ETL process. Rahm and Do (2000) further detail this phase in the following tasks: (a) data analysis, (b) definition of transformation workflow and mapping rules, (c) verification, (d) transformation, and (e) backflow of cleaned data.

In terms of the transformation tasks, we can categorize the problem in two main classes of problems (Lenzerini, 2002):

- conflicts and problems at the schema level, and
- data-level transformations (i.e., at the instance level).

The main problems with the schema level are (a) naming conflicts, where the same name is used for different objects (homonyms) or different names are used for the same object (synonyms), and (b) structural conflicts, where one must deal with different representations of the same object in different sources.

In addition, there are many variations of data-level conflicts across sources: duplicated or contradicting records, different value representations (e.g., marital status), different interpretation of the values (e.g., measurement units dollar vs. euro), different aggregation levels (e.g., sales per product vs. sales per product group), or reference to different points in time (e.g., current sales as of yesterday for source 1 vs. current sales as of last week for source 2). The list is enriched by low-level technical problems such as data type conversions, applying format masks, assigning fields to a sequence number, substituting constants, setting values to NULL or DEFAULT based on a condition, or using simple SQL operators (e.g., UPPER, TRUNC, SUBSTR).

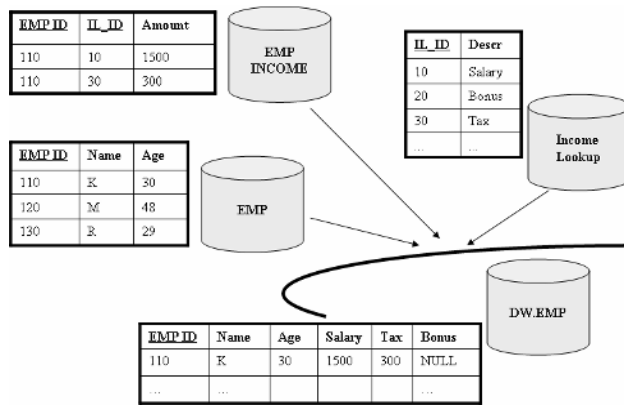
In sequel, we present three common ETL transformations as examples: (a) semantic reconciliation and denormalization; (b) surrogate key assignment; and (c) string problems.

- **Semantic reconciliation and denormalization:** Frequently, DWs are highly denormalized, to answer more quickly certain queries. For example, in Figure 4, one can observe that a query on the total income of an employee in table DW.EMP can easily be computed as an addition of the attributes Salary, Tax, Bonus, whereas in the schema of the OLTP table EMP\_INCOME, we should apply an aggregation operation. The transformation of the information organized in rows to the information organized in columns is called rotation or denormalization, because, frequently, the derived values (e.g., the total income) are also



## Extraction-Transformation-Loading Processes

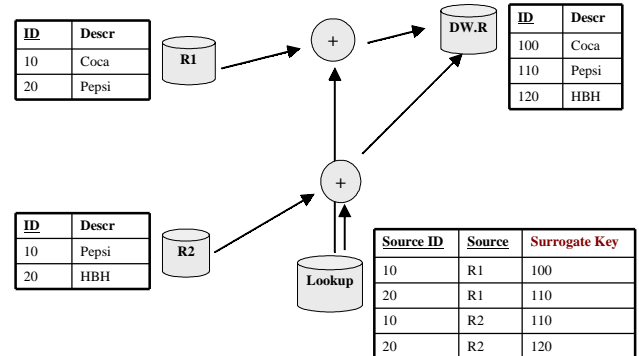
Figure 4. Semantic reconciliation and denormalization



stored as columns, functionally dependent on other attributes. Occasionally, it is possible to apply the reverse transformation to normalize denormalized data before being loaded to the DW. Observe also, how the different tables at the source side (i.e., EMP, EMP\_INCOME, Income\_Lookup) are integrated into a single DW table (i.e., DW.EMP).

- Surrogate keys:** In a DW project, we usually replace the keys of the production systems with a uniform key, which we call a *surrogate key* (Kimball, Reeves, Ross, & Thornthwaite, 1998). Reasons for this replacement are performance and semantic homogeneity. Performance is affected because textual attributes are not the best candidates for indexed keys and thus need to be replaced by integer keys. More important, semantic homogeneity causes reconciliation problems because different production systems might use different keys for the same object (synonyms), or the same key for different objects (homonyms), resulting in the need for a global replacement of those values in the DW. Observe row 10, Pepsi, in table R2 of Figure 5. This row has a synonym conflict with row 20, Pepsi, in table R1 because they both represent the same real-world entity with different IDs, and it has a homonym conflict with row 10, Coca, in table R1 (over attribute ID). The production key ID is replaced by a surrogate key through a lookup table of the form Lookup(SourceID,Source,SurrogateKey). The Source column of this table allows that there can be synonyms in the different sources, which are mapped to different objects in the DW (e.g., value 10 in tables R1 and R2). At the end of this process, the DW table DW.R has globally unique, reconciled keys.

Figure 5. Surrogate key assignment



- String problems:** A major challenge in ETL processes is the cleaning and homogenization of string data, that is, data that stands for addresses, acronyms, names, and so forth. Usually, the approaches for the solution of this problem include the application of regular expressions (e.g., using Perl programs) for the normalization of string data to a set of reference values. Figure 6 illustrates an example of this problem.

## Loading

The final loading of the DW has its own technical challenges. Simple SQL commands are not sufficient because the open-loop-fetch technique, in which records are inserted one by one, is extremely slow for the vast volume of data to be loaded in the warehouse. An extra problem is the simultaneous usage of the rollback segments and log files during the loading process. Turning them off might include some risk in the case of a loading failure. So far, the best technique is the usage of the batch loading tools offered by most RDBMSs that avoids these problems.

Another problem is discriminating between new and existing data at loading time. This problem arises when a

Figure 6. String problems

Source Value	DW value
HP	HP
H.P.	HP
H-P	HP
Hewlett-Packard	HP
Hioulet-Pakard	HP
DEC	DEC
Digital Co.	DEC
...	...

set of records has to be classified to (a) the new rows that need to be appended to the warehouse and (b) rows that already exist in the DW but whose value has changed and must be updated (e.g., with an UPDATE command). Modern ETL tools already provide mechanisms towards this problem, mostly through language predicates.

Techniques that facilitate the loading task involve the simultaneous creation of tables and their respective indexes, the minimization of interprocess wait states, and the maximization of concurrent CPU usage.

## FUTURE TRENDS

In a recent study, Giga Information Group (2002) reported that the ETL market reached a size of \$667 million for year 2001; still, the growth rate reached a rather low 11% (compared with a 60% growth rate for year 2000). The study also proposed future technological challenges and forecasts that involve the integration of ETL with (a) XML adapters; (b) EAI (Enterprise Application Integration) tools (e.g., MQ-Series); (c) customized data quality tools; and (d) the move towards parallel processing of the ETL workflows.

There are several issues that are technologically open and that present interesting topics of research for the future. Active ETL, the need to refresh the warehouse with the freshest data possible (ideally, online) is a hot topic that has recently appeared (Adzic & Fiore, 2003). The need for optimal algorithms, for the individual transformations and for the whole process, is also an interesting topic of research. Finally, we anticipate that the extension of the ETL mechanisms for nontraditional data such as XML/HTML, spatial and biomedical data will also be a hot topic of research.

## CONCLUSION

ETL tools are pieces of software responsible for the extraction of data from several sources, their cleansing, customization, and insertion into a DW. In all the phases of an ETL process (extraction and exportation, transformation and cleaning, and loading), individual issues arise and, along with the problems and constraints that concern the overall ETL process, make its lifecycle very troublesome. Although, state of the art in the field of both research and commercial ETL tools includes some pieces of remarkable progress, a lot of work remains to be done before we claim that this problem is resolved. To this end, recent studies account this subject a research challenge and pinpoint the main topics of future work.

## REFERENCES

- Adzic, J., & Fiore, V. (2003). Data warehouse population platform. *Proceedings of 5th International Workshop on the Design and Management of Data Warehouses (DMDW '03)*, Berlin, Germany.
- Borkar, V., Deshmuk, K., & Sarawagi, S. (2000). Automatically extracting structure from free text addresses. *Bulletin of the Technical Committee on Data Engineering*, 23(4).
- Calì, A., Calvanese, D., De Giacomo, G., Lenzerini, M., Naggari, P., & Vernacotola, F. (2003). IBIS: Semantic data integration at work. *Proceedings of the 15th CAiSE: Vol. 2681. Lecture Notes in Computer Science* (pp. 79-94). Springer.
- Galhardas, H., Florescu, D., Shasha, D., & Simon, E. (2000). Ajax: An extensible data cleaning tool. In *Proceedings of the ACM SIGMOD international conference on the management of data* (p. 590). Dallas: TX.
- Giga Information Group. (2002). *Market overview update: ETL* (Tech. Rep. No. RPA-032002-00021).
- Inmon, W.-H. (1996). *Building the data warehouse* (2nd ed.). New York: John Wiley & Sons.
- Jarke, M., Lenzerini, M., Vassiliou, Y., & Vassiliadis, P. (Eds.). (2000). *Fundamentals of data warehouses*. Springer-Verlag.
- Kimball, R., Reeves, L., Ross, M., & Thornthwaite, W. (1998). *The data warehouse lifecycle toolkit: Expert methods for designing, developing, and deploying data warehouses*. John Wiley & Sons.
- Labio, W., & Garcia-Molina, H. (1996). Efficient snapshot differential algorithms for data warehousing. *Proceedings of the 22nd international conference on very large data bases (VLDB)* (pp. 63-74). Bombay, India.
- Lenzerini, M. (2002). Data integration: A theoretical perspective. *Proceedings of the 21st symposium on principles of database systems (PODS)* (pp. 233-246). Wisconsin.
- Monge, A. (2000). Matching algorithms within a duplicate detection system. *Bulletin of the Technical Committee on Data Engineering*, 23(4).
- Rahm, E., & Do, H. H. (2000). Data cleaning: Problems and current approaches. *Bulletin of the Technical Committee on Data Engineering*, 23(4).
- Raman, V., & Hellerstein, J. (2001). Potter's Wheel: An interactive data cleaning system. *Proceedings of the 27th international conference on very large data bases (VLDB)* (pp. 381-390). Rome, Italy.

Rundensteiner, E. (Ed.). (1999). Data transformations [Special issue]. *Bulletin of the Technical Committee on Data Engineering*, 22(1).

Sarawagi, S. (2000). Data cleaning [Special issue]. *Bulletin of the Technical Committee on Data Engineering*, 23(4).

Scalzo, B. (2003). *Oracle DBA guide to data warehousing and star schemas*. Prentice Hall.

Simitsis, A. (2004). List of ETL tools. Retrieved May 10, 2004, from <http://www.dbnet.ece.ntua.gr/~asimi/ETLTools.htm>

Simitsis, A., Vassiliadis, P., Skiadopoulos, S., & Sellis, T. (2003). Modeling of ETL processes using graphs. *Proceedings of the 2nd Hellenic Data Management Symposium (HDMS03)*, Athens, Greece.

Simitsis, A., & Vassiliadis, P. (2003). A methodology for the conceptual modeling of ETL processes. *Proceedings of the decision systems engineering (DSE '03)*, Velden, Austria.

Vassiliadis, P., Simitsis, A., Georgantas, P., Terrovitis, M., & Skiadopoulos, S. (2004). A generic and customizable framework for the design of ETL scenarios. *Information Systems Journal*.

Vassiliadis, P., Simitsis, A., & Skiadopoulos, S. (2002). Conceptual modeling for ETL processes. *Proceedings of the 5th data warehousing and OLAP (DOLAP '02)*, McLean, VA.

## KEY TERMS

**Data Mart:** A logical subset of the complete data warehouse. We often view the data mart as the restriction of the data warehouse to a single business process or to a group of related business processes targeted towards a particular business group.

**Data Staging Area (DSA):** An auxiliary area of volatile data employed for the purpose of data transformation, reconciliation, and cleaning before the final loading of the data warehouse.

**Data Warehouse (DW):** A data warehouse is a subject-oriented, integrated, time-variant, nonvolatile collection of data used to support the strategic decision-making processes for the enterprise. It is the central point of data integration for business intelligence and is the source of data for the data marts, delivering a common view of enterprise data (Inmon, 1996).

**ETL:** Extract, transform, and load (ETL) are data warehousing functions that involves extracting data from outside sources, transforming it to fit business needs, and ultimately loading it into the data warehouse. ETL is an important part of data warehousing; it is the way data actually gets loaded into the warehouse.

**On-Line Analytical Processing (OLAP):** The general activity of querying and presenting text and number data from data warehouses as well as a specifically dimensional style of querying and presenting that is exemplified by a number of OLAP vendors.

**Source System:** An operational system of record whose function is to capture the transactions of the business. A source system is often called a legacy system in a mainframe environment.

**Target System:** The physical machine on which the data warehouse is organized and stored for direct querying by end users, report writers, and other applications.

# Free Software and Open Source Databases

**Hugo J. Curti**

*Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina*

## INTRODUCTION

Free software and open source databases have become a serious alternative to commercial databases. This article states common points and differences between the open source and free software movements and briefly describes the most relevant licenses and database projects.

## BACKGROUND: FREE SOFTWARE AND OPEN SOURCE

### Free Software

There are many ambiguities involved in the definition of free software. The word “free” may be used to say “free of charge” or to say “free speech.” It is the same word, but the meaning is different. When it is used in “Free Software,” *free* means *freedom*. Freedom to run a program. Freedom to study how a program works and adapt it to specific needs. Freedom to redistribute a program (even charging money for that). Freedom to improve a program and redistribute the improvements. A pure free software distribution license should grant all those rights to users and should also forbid any project using the software to revoke those rights. When the license states that “every project that uses this software in any form should be released as free software,” it is said to be a “copylefted” license. There are moral reasons to argue that every free software should be copylefted, although many weak copylefted or even non-copylefted free software licenses exist.

There are also many technical advantages of free software over closed or proprietary software. As many developers can study the sources, most bugs are discovered at the beginning of free software’s life cycle, tending to stabilize it faster. Free software projects are generally based on relatively simple modules that are combined to form a complex project, facilitating in this way software reuse and drastically reducing the programming efforts. Somehow it can be said that those problems that are solved synergically in the free software world are solved, on the other hand, by means of man hours and money in the proprietary software world.

The biggest free software project in the world is the GNU<sup>1</sup> project, whose main sponsor is the FREE SOFTWARE FOUNDATION. There are other important free software sponsors around the world (Association for Free Software, 2004; Free Software Foundation Europe, 2004; Ray, Hudson, & Greve, 2004). Summarizing, it can be said that free software is a big paradigm that includes technical, philosophical, moral, and social bases. There is a lot of information about free software provided by the Free Software Foundation (2004a).

### Open Source Initiative

The Open Source Initiative is a marketing program for free software (Debian Project, 2004; Open Source Initiative, 2004b; Perens, 1997).

It is based on more pragmatic reasons than free software: It emphasizes the technical benefits of providing the source code (full peer review and rapid evolution) over the ideological reasons. It was thought as way to let free software reach the corporate world. The Open Source Initiative provides a certification mark that may be used in any software distributed under the terms of an approved license.

### Free/Open Source Licenses

Only the most relevant licenses are listed here. The full list of Open Source licenses is published by the Open Source Initiative (2004a), and the full list of free software licenses is published by the Free Software Foundation (2004b).

### The GNU General Public License (GPL)

- Allows verbatim distribution of the source code.
- Allows distribution of binaries as long as the source code used to build it is available somehow.
- Allows modifications of the software to generate a new project, provided that the differences are clearly stated.
- Forbids the imposition of further restrictions in the distribution license of derived projects (“copyleft”).

- Provides no warranties of any kind, including the implied warranties of merchantability and fitness for a particular purpose.

### The New BSD License and the X11 (MIT) License

These are very simple licenses. Everything is permitted except to sue the authors of the code (Wheeler, 2004). There is also an old version of the BSD license, which has a clause that enforces advertising Berkeley University. This clause does not exist in the new BSD license.

### The Mozilla Public License

The Mozilla Public license is a non-strong copylefted license that has some complex restrictions that make it incompatible with the GNU GPL. That is, a module covered by the GPL and a module covered by the MPL cannot legally be linked together (Free Software Foundation, 2004b).

## FREE/OPEN SOURCE DATABASE PROJECTS

### PostgreSQL

POSTGRESQL (2004) is an object-relational database management system (ORDBMS). It is based on a research project at the University of California called POSTGRES. POSTGRES pioneered many concepts that only became available in some commercial database systems much later. Because of its extensibility, POSTGRESQL is also very well suited for database research.

#### Features

- Table inheritance.
- Complex queries support.
- Foreign keys and referential integrity support.
- Transactional integrity.
- Multiversion concurrency control.
- New procedural languages for stored procedures and triggers may be added by the user.
- New data types, functions, operators, aggregate functions, and index methods may be added by the user.

- Support for updatable views and query rewrite rules.
- Replication servers are under development. (erServer, 2004; Slony I, 2004).

#### License

POSTGRESQL is released under the terms of the BSD public license.

### MySQL

The MySQL software (2004) delivers a very fast, multi-threaded, multiuser, and robust SQL (Structured Query Language) database server. MySQL Server is intended for mission-critical, heavy-load production systems as well as for embedding into mass-deployed software.

#### Features

- Written in C and C++.
- Uses GNU Automake, Autoconf, and Libtool (the standard open source tools), enhancing the portability of the code.
- Fully multi-threaded.
- Transactional and non-transactional storage engine.
- The server is available as a separate program for network client/server operation or as a library for embedded stand-alone operation.
- APIs for C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, and Tcl are available.
- Partial stored procedures and foreign key support.

#### License

MYSQL has a dual license approach. Users may choose to use it under the terms of the GNU GPL or purchase a commercial license from MYSQL AB. The reference manual is not covered under the GNU GPL (an authorization from MYSQL AB is required to sell printed copies).

### Firebird

Firebird is an open source relational database that evolved from the commercial database INTERBASE by INPRISE, now known as BORLAND SOFTWARE CORP. (Firebird, 2004). In August 2000 this company released a beta version of Interbase 6.0 open source.



After that, a core of developers formed the FIREBIRD project based on it. Now the FIREBIRD database is almost 100% compatible with INTERBASE, although both are independent.

## Features

- Most ANSI SQL 92 features implemented.
- Runs on Windows, Linux, and a variety of Unix platforms.
- “Classic” architecture for single-threaded environments and “super server” for multi-threaded environments.
- Multiversion concurrency control.
- Various SQL dialects for backwards compatibility reasons.
- Native language support for stored procedures and triggers.
- Updatable views.
- Integrity check constraints.

## License

Firebird is released under the terms of the INTERBASE PUBLIC LICENSE, which is based on the MOZILLA PUBLIC LICENSE.

## FUTURE TRENDS AND CONCLUSION

Free software and open source (FS/OS) databases are becoming serious alternatives to big and expensive commercial databases. Many features present in commercial databases today were developed in FS/OS databases. As FS/OS projects evolve, databases get more complex and more complete.

Free software and open source are two definitions of a new paradigm for software development. There is a big gap between the corporate world and this paradigm, but the gap seems to be getting smaller and smaller in recent years (Yoder, 2002). According to the author’s personal opinion, chances are that in the near future this new paradigm will generate new business models and will replace the old ones.

## REFERENCES

Association for Free Software. (2004). *AFFS FAQs*. Retrieved from <http://www.affs.org.uk/faq.html>

Debian Project. (2004, April). *The Debian social contract, Version 1.0*. Retrieved from [http://www.debian.org/social\\_contract.html](http://www.debian.org/social_contract.html)

erServer. (2004). erServer main site. <http://www.erserver.com>

Firebird. (2004, October). Firebird project’s main site. <http://firebird.sourceforge.net>

Free Software Foundation. (2004a, October). *The GNU operating system*. Retrieved from <http://www.gnu.org>

Free Software Foundation. (2004b, October). *Various licenses and comments about them*. Retrieved from <http://www.gnu.org/licenses/license-list.html>

Free Software Foundation Europe. (2004, October). Free software foundation Europe main site. <http://www.fsfeurope.org>

MySQL. (2004, October). MySQL developer’s zone. <http://www.mysql.org>

Open Source Initiative. (2004a). The approved licenses. Retrieved from <http://www.opensource.org/licenses>

Open Source Initiative. (2004b). The Open Source Definition. Retrieved from <http://www.open source.org/docs/definition.php>

Perens, B. (1997, June). The Debian free software guidelines (DFSG). Retrieved from [http://www.debian.org/social\\_contract.html#guidelines](http://www.debian.org/social_contract.html#guidelines)

PostgreSQL. (2004, October). PostgreSQL project’s main site. <http://www.postgresql.org>

Ray, M. J., Hudson, A., & Greve, G. (2004). Constitution for AFFS. Retrieved from <http://www.affs.org.uk/documents/affs-const.txt>

Slony I. (2004). The slony i project main site. <http://www.slony.info>

Wheeler, D. A. (2004, September). Why open source software / free software (oss/fs)? Look at the numbers! Retrieved from [http://www.dwheeler.com/oss\\_fs\\_why.html](http://www.dwheeler.com/oss_fs_why.html)

Yoder, M. (2002, October). Our open source / free software future: It’s just a matter of time. Retrieved from <http://yoderdev.com/oss-future.html>

## KEY TERMS

**Closed (or Proprietary) Software:** Software that is owned by an individual or a company (usually the one that develops it) and is protected from unauthorized use by patents, trademarks, or copyrights. Such software is often sold or leased to other individuals or organizations, usually with strict restrictions regarding its use, modification, and further distribution. In many cases the source code is kept secret.

**Free Software Foundation (FSF):** Nonprofit organization founded in 1985 to support the GNU project and to promote education about free software.

**GNU Project:** Project launched by Richard Stallman with the goal of creating a complete free operating system: the GNU system. GNU is a recursive acronym for “GNU’s Not Unix.”

**Multi-Threaded System:** System where many concurrent sequences of executing instructions (called *threads*) may coexist and share data.

**Multiversion Concurrency Control:** Method used in relational databases to achieve serializability of transactions. MCC ensures a transaction never has to wait for a database object by maintaining several versions of an object. Each version would have a write time-stamp, and it would let a transaction read the most recent version of an object which precedes the transaction’s time stamp.

**Open Source Initiative (OSI):** An organization dedicated to managing and promoting the Open Source Definition for the good of the community.

**Software License:** Contract between a producer and a user of computer software, sometimes called an end-user license agreement (EULA), that specifies the parameters of the permission granted by the owner to the user.

## ENDNOTE

- <sup>1</sup> Recursive acronym meaning GNU is Not Unix

# Fuzzy Database Modeling

**Z.M.Ma**

*Northeastern University, China*

## INTRODUCTION

A major goal for database research has been the incorporation of additional semantics into the data model. Classical data models often suffer from their incapability of representing and manipulating imprecise and uncertain information that may occur in many real-world applications. Since the early 1980s, Zadeh's fuzzy logic (Zadeh, 1965) has been used to extend various data models. The purpose of introducing fuzzy logic in database modeling was to enhance the classical models such that uncertain and imprecise information can be represented and manipulated. This resulted in numerous contributions, mainly with respect to the popular relational model or to some related form of it.

Also, rapid advances in computing power have brought opportunities for databases in emerging applications in CAD/CAM (Computer-Aided Design/Computer-Aided Manufacturing), multimedia, and geographic information systems (GIS). These applications characteristically require the modeling and manipulation of complex objects and semantic relationships. It proved that the object-oriented paradigm lends itself extremely well to the requirements. Because the classical relational database model and its extension of fuzziness do not satisfy the need of modeling complex objects with imprecision and uncertainty, many current researches have concentrated on fuzzy object-oriented database models to deal with complex objects and uncertain data together.

## BACKGROUND

Database modeling can be carried out at two different levels: conceptual data modeling and database modeling. Therefore, we have conceptual data models (e.g., ER/EER—Entity-Relationship/Extended Entity-Relationship and UML) and logical database models (relational databases and object-oriented databases). Logical database models are often created through mapping conceptual data models into logical database models. This conversion is called *conceptual design of databases*.

In order to deal with imprecise and uncertain information in database modeling, fuzzy set has been applied. Let  $U$  be a universe of discourse, then a fuzzy value on  $U$

is characterized by a fuzzy set  $F$  in  $U$ . A membership function  $\mu_F: U \rightarrow [0, 1]$  is defined for the fuzzy set  $F$ , where  $\mu_F(u)$ , for each  $u \in U$ , denotes the degree of membership of  $u$  in the fuzzy set  $F$ . Thus the fuzzy set  $F$  is described as follows.

$$F = \{\mu(u_1)/u_1, \mu(u_2)/u_2, \dots, \mu(u_n)/u_n\}$$

When  $\mu_F(u)$  is viewed to be a measure of the possibility that a variable  $X$  has the value  $u$  in this approach, where  $X$  takes values in  $U$ , a fuzzy value is described by a possibility distribution  $\pi_X$  (Zadeh, 1978).

$$\pi_X = \{\pi_X(u_1)/u_1, \pi_X(u_2)/u_2, \dots, \pi_X(u_n)/u_n\}$$

Here,  $\pi_X(u_i)$ ,  $u_i \in U$ , denotes the possibility that  $u_i$  is true. Let  $\pi_X$  and  $F$  be the possibility distribution representation and the fuzzy set representation for a fuzzy value, respectively. It is apparent that  $\pi_X = F$  is true (Raju & Majumdar, 1988).

Also, fuzzy data can be represented by similarity relations in domain elements (Buckles & Petry, 1982), in which the fuzziness comes from the similarity relations between two values in a universe of discourse, not from the status of an object itself. Similarity relations are thus used to describe the degree similarity of two values from the same universe of discourse. A similarity relation  $Sim$  on the universe of discourse  $U$  is a mapping:  $U \times U \rightarrow [0, 1]$  such that:

- a. for  $\forall x \in U, Sim(x, x) = 1$ , (reflexivity)
- b. for  $\forall x, y \in U, Sim(x, y) = Sim(y, x)$ , and (symmetry)
- c. for  $\forall x, y, z \in U, Sim(x, z) \geq \max_y(\min(Sim(x, y), Sim(y, z)))$ . (transitivity)

## MAJOR ISSUES AND SOLUTIONS

### Fuzzy Relational Databases

Fuzzy information has been extensively investigated in the context of the relational databases. The followings are some major issues in current studies of fuzzy relational databases (Ma & Mili, 2002b):

## Fuzzy Relational Database Models

One can find several kinds of fuzzy relational database models. One of the fuzzy relational data models is based on similarity relations (Buckles & Petry, 1982), or proximity relation (Shenoi & Melton, 1989), or resemblance (Rundensteiner, Hawkes, & Bandler, 1989). The other one is based on possibility distribution (Prade & Testemale, 1984), which can further be classified into two categories: tuples associated with possibilities and attribute values-represented possibility distributions (Raju & Majumdar, 1988). The form of an  $n$ -tuple in each of the previously mentioned fuzzy relational model can be expressed, respectively, as:

$$t = \langle p_1, p_2, \dots, p_i, \dots, p_n \rangle, t = \langle a_1, a_2, \dots, a_i, \dots, a_n, d \rangle$$

$$\text{and } t = \langle \pi_{A_1}, \pi_{A_2}, \dots, \pi_{A_i}, \dots, \pi_{A_n} \rangle,$$

where  $p_i \subseteq D_i$  with  $D_i$  being the domain of attribute  $A_i$ ,  $a_i \in D_i$ ,  $d \in (0, 1)$ ,  $\pi_{A_i}$  is the possibility distribution of attribute  $A_i$  on its domain  $D_i$ , and  $\pi_{A_i}(x)$ ,  $x \in D_i$ , denotes the possibility that  $x$  is the actual value of  $t[A_i]$ .

It is clear that one can combine two kinds of fuzziness in possibility-based fuzzy relational databases, where attribute values may be possibility distributions and tuples are connected with membership degrees. Such fuzzy relational databases are called possibility-distribution-fuzzy relational models (Umano & Fukami, 1994). Another possible extension is to combine possibility distribution and similarity (proximity or resemblance) relation, and the extended possibility-based fuzzy relational databases are hereby proposed (G. Q. Chen, Vandenbulcke, & Kerre, 1992; G. Q. Chen, Kerre, & Vandenbulcke, 1994, 1996; Ma & Mili, 2002; Ma, Zhang, & Ma, 2000; Ma, Zhang, & Mili, 2002), where possibility distribution and resemblance relation arise in a relational databases simultaneously.

## Fuzzy Data Integrity Constraints and Formalizations

Fuzzy data dependencies, mainly including fuzzy functional dependency (FFD) and fuzzy multivalued dependency (FMVD), have extensively been studied in the context of fuzzy relational databases. There are some papers that focus only on FMVD (Bhattacharjee & Mazumdar, 1998; Jyothi & Babu, 1997; Tripathy & Sakena, 1990; ). And some papers focus only on FFD, where we can classify two kinds of papers: The first one focuses on the axiomatization of FFD (G. Q. Chen, Kerre, & Vandenbulcke, 1994, 1995; Cubero & Vila, 1994; Liao, Wang, & Liu, 1999; Liu, 1992, 1993a, 1993b; Saxena & Tyagi, 1995) and the second focuses on the lossless join and decomposition (Bhuniya &

Niyogi, 1993; Bosc & Pivert, 2003; Raju & Majumdar, 1988). The later is the basis on which to implement the normalization of fuzzy relational databases (G. Q. Chen, Kerre, & Vandenbulcke, 1996). Also, there are some papers that focus both on FFD and FMVD and present the axiomatization of FFD and FMVD (Liu, 1997; Ma, Zhang, Ma, & Mili, 2002; Sözat & Yazici, 2001; Yazici & Sözat, 1998).

In addition, fuzzy data dependencies can be applied in data handling. In Bosc, Dubois, and Prade (1998), FFD was used for redundancy elimination. In Intan and Mukaidono (2000), FFD was used for approximate data querying. In Chang and Chen (1998), Liao, Wang, and Liu (1999), and Ma, Zhang, and Mili (2002), FFD was used for fuzzy data compression.

## Query and Data Processing

Classical relational databases suffer from a lack of flexibility in query. The given selection condition and the contents of the relation are all crisp. A query is flexible if the following conditions can be satisfied (Bosc & Pivert, 1992):

- A qualitative distinction between the selected tuples is allowed.
- Imprecise conditions inside queries are introduced when the user cannot define his or her needs in a definite way, or when a prespecified number of responses are desired and therefore a margin is allowed to interpret the query.

Here, typically, the former case occurs when the queried relational databases contain incomplete information and the query conditions are crisp, and the later case occurs when the query conditions are imprecise, even if the queried relational databases do not contain incomplete information.

Takahashi presented a fuzzy query language for relational databases (Takahashi, 1991) and fuzzy database query languages and their relational completeness theorem (Takahashi, 1993). Bosc and Lietard (1996) presented the concepts of fuzzy integrals and database flexible querying. Bosc and Pivert (1995) presented a relational database language called SQLf for fuzzy querying. Also, S. M. Chen and Jong (1997) and Y. C. Chen and Chen (2000) presented fuzzy query translation techniques for relational database systems and techniques of fuzzy query processing for fuzzy database systems, respectively. In addition, based on matching strengths of answers in fuzzy relational databases, Chiang, Lin, and Shis (1998) presented a method for fuzzy query processing. Yang et al. (2001) focused on nested fuzzy SQL queries in a fuzzy relational database.

Fuzzy logic techniques have been used in multimedia database querying (Dubois, Prade, & Sedes, 2001). In Kacprzyk, Zadrozny, and Ziolkowski, (1987), a “human-consistent” database querying system based on fuzzy logic with linguistic quantifiers was presented. Using clustering techniques, Kamel, Hatfield, and Ismail (1990) presented a fuzzy query processing method.

In addition to query processing in fuzzy relational databases, there are also studies focusing on the operations of relational algebra in fuzzy relational databases (Ma & Mili, 2002; Umano & Fukami, 1994). In Zhang and Wang (2000), a type of fuzzy equi-join was defined using fuzzy equality indicators.

## Fuzzy Nested Relational Databases

In Yazici et al. (1999), an extended nested relational data model (also known as an  $NF^2$  data model) was introduced for representing and manipulating complex and uncertain data in databases, and the extended algebra and the extended SQL-like query language were hereby defined. Also, physical data representation of the model and the core operations that the model provides were also introduced. Ma and Mili (2002) based possibility distribution rather than the similarity relations in Yazici et al. (1999), an extended possibility-based fuzzy nested relational database model was introduced and its algebra is hereby developed.

It should be noted that the  $NF^2$  data model is able to handle complex-valued attributes and may be better suited to some complex applications such as office automation systems, information retrieval systems, and expert database systems. But it is difficult for  $NF^2$  data model to represent complex relationships among objects and attributes. Some advanced abstracts in data modeling (e.g., class hierarchy, inheritance, superclass/subclass, and encapsulation), which are needed by many real applications, are not supported by the  $NF^2$  data model. Therefore, in order to model uncertain data and complex-valued attributes as well as complex relationships among objects, current efforts have focused on conceptual data models and object-oriented databases (OODB) with imprecise and uncertain information.

## Fuzzy Object-Oriented Databases

The incorporation of imprecise and uncertain information in object-oriented databases has increasingly received attention where fuzziness is witnessed at the levels of object instances and class hierarchies. Based on similarity relationship, George, Srikanth, Petru, and Buckles, 1996 used the range of attribute values to represent the set of allowed values for an attribute of a given class. Depending on the inclusion of the actual

attribute values of the given object into the range of the attributes for the class, the membership degrees of an object to a class can be calculated. The weak and strong class hierarchies were defined based on monotone increase or decrease of the membership of a subclass in its superclass. Design and implementation issues in such a fuzzy object-oriented data model were presented in Yazici, George, and Aksoy (1998).

A UFO (i.e., uncertainty and fuzziness in an object-oriented databases model) was proposed by Van Gyseghem and Caluwe (1998) to model fuzziness and uncertainty by means of fuzzy set theory and generalized fuzzy set, respectively. That the behaviour and structure of the object are incompletely defined results in a gradual nature for the instantiation of an object. The partial inheritance, conditional inheritance, and multiple inheritances are permitted in fuzzy hierarchies.

In Lee, Xue, Hsu, and Yang, (1999), an approach to OO (object-oriented) modeling based on fuzzy logic is proposed to formulate imprecise requirements along four dimensions: fuzzy class, fuzzy rules, fuzzy class relationships, and fuzzy associations between classes.

Based on the extension of a graph-based model object model, a fuzzy object-oriented data model was defined in Bordogna, Pasi, and Lucarella (1999). The notion of strength expressed by linguistic qualifiers was proposed, which can be associated with the instance relationship as well as an object with a class. Fuzzy classes and fuzzy class hierarchies were thus modeled in the OODB. The definition of graph-based operations to select and browse such a fuzzy object-oriented database that manages both crisp and fuzzy information was proposed in Bordogna and Pasi (2001).

Based on possibility theory, vagueness and uncertainty were represented in class hierarchies in Dubois, Prade, and Rossazza (1991), where the fuzzy ranges of the subclass attributes defined restrictions on that of the superclass attributes, and then the degree of inclusion of a subclass in the superclass was dependent on the inclusion between the fuzzy ranges of their attributes. Also based on possibility distribution theory, Ma, Zhang, and Ma (2004) extended under the fuzzy information environment some major notions in object-oriented databases such as objects, classes, objects-classes relationships, subclass/superclass, and multiple inheritances. A generic model for fuzzy object-oriented databases and some operations were hereby developed.

In addition to major focus on fuzzy object-oriented database model, there are studies that mainly concentrate on issues related to fuzzy object-oriented databases. Fuzzy types were added into fuzzy object-oriented databases to manage vague structures in Marín,



Vila, and Pons (2000) and Marín, Pons, and Vila (2001). And in Marín, Medina, Pons, Sánchez, and Vila (2003), complex object comparison in a fuzzy context was developed. In Cross (2001, 2003), fuzzy relationships in object models were investigated.

Also, some efforts have been paid to the establishment of a consistent framework for a fuzzy object-oriented model based on the standard for the object data management group (ODMG) object data model (Cross, Caluwe, & Van Gyseghem, 1997). And some special fuzzy object-oriented databases (e.g., fuzzy deductive object-oriented databases; Yazici & Koyuncu, 1997; Koyuncu & Yazici, 2003), fuzzy and probabilistic object bases (Cao & Rossiter, 2003) have been developed. In addition, fuzzy object-oriented database have been applied in some areas such as GIS (Cross & Firat, 2000) and multimedia (Majumdar, Bhattacharya, & Saha, 2002). Concerning most recent research and application issues about fuzzy object-oriented databases, refer to Ma (2005).

## Fuzzy Conceptual Data Models and Fuzzy Database Design

### Fuzzy Conceptual Data Models

Based on fuzzy set theory, Zvieli and Chen (1986) introduced three levels of fuzziness in the ER model. At the first level, entity sets, relationships, and attribute sets may be fuzzy, namely, they have a degree of membership in the model. The second level is related to the fuzzy occurrences of entities and relationships. The third level concerns the fuzzy values of attributes of special entities and relationships. By using fuzzy set theory, the fuzzy extensions of several major EER concepts were introduced in G. Q. Chen and Kerre (1998), including superclass/subclass, generalization/specialization, category, and the subclass with multiple superclasses.

In addition to the ER/EER model, the IFO data model (Abiteboul & Hull, 1987) is a mathematically defined conceptual data model that incorporates the fundamental principles of semantic database modeling within a graph-based representational framework. The extensions of the IFO to deal with fuzzy information were proposed in Vila et al., 1996). In (Vila, Cubero, Medina, & Pons, 1996), several types of imprecision and uncertainty, such as the values without semantic representation, the values with semantic representation and disjunctive meaning, the values with semantic representation and conjunctive meaning, and the representation of uncertain information, were incorporated into the attribute domain of the object-based data model. In addition

to the attribute-level uncertainty, the uncertainty was also considered to be at the object and class level.

## Conceptual Design of Fuzzy Databases

Traditional databases are generally designed from conceptual data models. By mapping, conceptual data models are converted into database models. It is shown that less research has been done on modeling fuzzy information in the conceptual data model. It is particularly true in developing design methodologies for implementing fuzzy databases.

In Chaudhry, Moyne, & Rundensteiner (1999), the fuzzy relational databases were designed by using the fuzzy ER model proposed in Zvieli and Chen (1986). In Yazici, Buckles, and Petry (1999), based on similarity relations (Buckles & Petry, 1982), the IFO (Abiteboul and Hull, 1987) model was extended to the *ExIFO* (Extended *IFO*) model to represent uncertainties at the levels of the attribute, the object, and class. Also, a mapping process to transform the *ExIFO* model into the fuzzy extended  $NF^2$  relations, including uncertain properties that are represented in both models, was also described. In Ma, Zhang, Ma, and Chen (2001), a full fuzzy EER model and the graphical representations were presented and the formal design methodology for fuzzy object-oriented databases from a fuzzy entity-relationship model was provided.

## FUTURE TRENDS

Compared with fuzzy relational databases, the research on fuzzy conceptual data models and fuzzy object-oriented databases is receiving increasing attention (Ma, 2005). In addition, the incorporation of probabilistic uncertain information into fuzzy data models should be interesting. Finally, building FDBMS and developing prototype systems of fuzzy databases are important topics in the area of fuzzy databases.

## CONCLUSION

This article reviews fuzzy database modeling technologies, including fuzzy conceptual data models and database models. Concerning fuzzy database models, fuzzy relational databases and fuzzy object-oriented databases are discussed, respectively. In fuzzy relational databases, the issues of data presentation and model structure, data dependencies and formalization, and data query and processing have extensively studied.

A major goal for database research has been the incorporation of additional semantics into the data model. Fuzzy database modeling can support imprecise and uncertain data storage, processing, and retrieval activities related to data management in intelligent information systems.

## REFERENCES

- Abiteboul, S., & Hull, R. (1987). IFO: A formal semantic database model. *ACM Transactions on Database Systems*, 12(4), 525-565.
- Bhattacharjee, T. K., & Mazumdar, A. K. (1998). Axiomatization of fuzzy multivalued dependencies in a fuzzy relational data model. *Fuzzy Sets and Systems*, 96(3), 343-352.
- Bhuniya, B., & Niyogi, P. (1993). Lossless join property in fuzzy relational databases. *Data and Knowledge Engineering*, 11(2), 109-124.
- Bordogna, G., Pasi, G., & Lucarella, D. (1999). A fuzzy object-oriented data model for managing vague and uncertain information. *International Journal of Intelligent Systems*, 14, 623-651.
- Bordogna, G., & Pasi, G. (2001). Graph-based interaction in a fuzzy object oriented database. *International Journal of Intelligent Systems*, 16(7), 821-841.
- Bosc, P., Dubois, D., & Prade, H. (1998). Fuzzy functional dependencies and redundancy elimination. *Journal of the American Society for Information Science*, 49(3), 217-235.
- Bosc, P., & Lietard, L. (1996). Fuzzy integrals and database flexible querying. *Proceedings of the 5th IEEE International Conference on Fuzzy Systems*, New Orleans, USA, (pp. 100-106).
- Bosc, P., & Pivert, O. (1992). Some approaches for relational databases flexible querying. *Journal of Intelligent Information Systems*, 1, 323-354.
- Bosc, P., & Pivert, O. (1995). SQLf: A relational database language for fuzzy querying *IEEE Transactions on Fuzzy Systems*, 3, 1-17.
- Buckles, B. P., & Petry, F. E. (1982) A fuzzy representation of data for relational database. *Fuzzy Sets and Systems*, 7(3), 213-226.
- Cao, T. H., & Rossiter, J. M. (2003). A deductive probabilistic and fuzzy object-oriented database language. *Fuzzy Sets and Systems*, 140, 129-150.
- Chang, C. S., & Chen, A. L. P. (1998). Efficient refinement of uncertain data by fuzzy integrity constraints. *Information Sciences*, 104(3/4), 191-211.
- Chaudhry, N. A., Moyne, J. R., & Rundensteiner, E. A. (1999). An extended database design methodology for uncertain data management. *Information Sciences*, 121(1/2), 83-112.
- Chen, G. Q., Kerre, E. E., & Vandenbulcke, J. (1994). A computational algorithm for the ffd closure and a complete axiomatization of fuzzy functional dependency (FFD). *International Journal of Intelligent Systems*, 9, 421-439.
- Chen G. Q., Kerre, E. E., & Vandenbulcke, J. (1995). The dependency-preserving decomposition and a testing algorithm in a fuzzy relational data model. *Fuzzy Sets and Systems*, 72(1), 27-37.
- Chen, G. Q., Kerre, E. E., & Vandenbulcke, J. (1996). Normalization based on functional dependency in a fuzzy relational data model. *Information Systems*, 21(3), 299-310.
- Chen, G. Q., Vandenbulcke, J., & Kerre, E. E. (1992). A general treatment of data redundancy in a fuzzy relational data model. *Journal of the American Society of Information Science*, 43(4), 304-311.
- Chen, G. Q., & Kerre, E. E. (1998). Extending ER/EER concepts towards fuzzy conceptual data modeling. *Proceedings of the 1998 IEEE International Conference on Fuzzy Systems*, 2, 1320-1325.
- Chen, S. M., & Jong, W. T. (1997). Fuzzy query translation for relational database systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 27, 714-721.
- Chen, Y. C., & Chen, S. M. (2000). Techniques of fuzzy query processing for fuzzy database systems. *Proceedings of the 5th Conference on Artificial Intelligence and Applications*, Taiwan, China (pp. 361-368).
- Chiang, D. A., Lin, N. P., & Shis, C. C. (1998). Matching strengths of answers in fuzzy relational databases. *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, 28, 476-481.
- Cross, V. (2001). fuzzy extensions for relationships in a generalized object model. *International Journal of Intelligent Systems*, 16(7), 843-861.
- Cross, V. (2003). Defining fuzzy relationships in object models: Abstraction and interpretation. *Fuzzy Sets and Systems*, 140, 5-27.
- Cross, V., Caluwe, R., & Van Gyseghem, N. (1997). A perspective from the fuzzy object data management group

- (FODMG). *Proceedings of the 1997 IEEE International Conference on Fuzzy Systems* (Vol. 2, pp. 721-728).
- Cross, V., & Firat, A. (2000). Fuzzy objects for geographical information systems. *Fuzzy Sets and Systems*, 113, 19-36.
- Cubero, J. C., & Vila, M. A. (1994). A new definition of fuzzy functional dependency in fuzzy relational databases. *International Journal of Intelligent Systems*, 9(5), 441-448.
- Dubois, D., Prade, H., & Rossazza, J. P. (1991). Vagueness, typicality, and uncertainty in class hierarchies. *International Journal of Intelligent Systems*, 6, 167-183.
- Dubois, D., Prade, H., & Sedes, F. (2001). Fuzzy logic techniques in multimedia database querying: A preliminary investigation of the potentials. *IEEE Transactions on Knowledge and Data Engineering*, 13, 383-392.
- Intan, R., & Mukaidono, M. (2000). Fuzzy functional dependency and its application to approximate data querying. *Proceedings of the 2000 International Database Engineering and Applications Symposium* (pp. 47-54).
- Jyothi, S., & Babu, M. S. (1997). Multivalued dependencies in fuzzy relational databases and lossless join decomposition. *Fuzzy Sets and Systems*, 88(3), 315-332.
- Kacprzyk, J., Zadrozny, S., & Ziolkowski, A. (1987). FQUERY III+: A "human-consistent" database querying system based on fuzzy logic with linguistic quantifiers. *Proceedings of the Second International Fuzzy Systems Association Congress*, Seattle, WA (pp. 443-453).
- Kamel, M., Hadfield, B., & Ismail, M. (1990). Fuzzy query processing using clustering techniques. *Information Processing and Management*, 26, 279-293.
- Koyuncu, M., & Yazici, A. (2003). IFOOD: An intelligent fuzzy object-oriented database architecture. *IEEE Transactions on Knowledge and Data Engineering*, 15(5), 1137-1154.
- Lee, J., Xue, N. L., Hsu, K. H., & Yang, S. J. H. (1999). Modeling imprecise requirements with fuzzy objects. *Information Sciences*, 118(1/4), 101-119.
- Liao, S. Y., Wang, H. Q., & Liu, W. Y. (1999). Functional dependencies with null values, fuzzy values, and crisp values. *IEEE Transactions on Fuzzy Systems*, 7(1), 97-103.
- Liu, W. Y. (1992). The reduction of the fuzzy data domain and fuzzy consistent join. *Fuzzy Sets and Systems*, 51(1), 89-96.
- Liu, W. Y. (1993a). Extending the relational model to deal with fuzzy values. *Fuzzy Sets and Systems*, 60(2), 207-212.
- Liu, W. Y. (1993b). The fuzzy functional dependency on the basis of the semantic distance. *Fuzzy Sets and Systems*, 59, 173-179.
- Liu, W. Y. (1997). Fuzzy data dependencies and implication of fuzzy data dependencies. *Fuzzy Sets and Systems*, 92(3), 341-348.
- Ma, Z. M. (2005). *Advances in fuzzy object-oriented databases: Modeling and applications*. Hershey, PA: Idea Group Publishing.
- Ma, Z. M., Zhang, W. J., & Ma, W. Y. (2000). Semantic measure of fuzzy data in extended possibility-based fuzzy relational databases. *International Journal of Intelligent Systems*, 15(8), 705-716.
- Ma, Z. M., Zhang, W. J., & Ma, W. Y. (2004). Extending object-oriented databases for fuzzy information modeling. *Information Systems*, 29(5), 421-435.
- Ma, Z. M., Zhang, W. J., Ma, W. Y., & Chen, G. Q. (2001). Conceptual Design of fuzzy object-oriented databases utilizing extended entity-relationship model. *International Journal of Intelligent Systems*, 16(6), 697-711.
- Ma, Z. M., Zhang, W. J., Ma, W. Y., & Mili, F. (2002). Data dependencies in extended possibility-based fuzzy relational databases. *International Journal of Intelligent Systems*, 17(3), 321-332.
- Ma, Z. M., Zhang, W. J., & Mili, F. (2002). Fuzzy Data compression based on data dependencies. *International Journal of Intelligent Systems*, 17(4), 409-426.
- Ma, Z. M., & Mili, F. (2002). Handling fuzzy information in extended possibility-based fuzzy relational databases. *International Journal of Intelligent Systems*, 17(10), 925-942.
- Ma, Z. M., & Mili, F. (2002b). An extended possibility-based fuzzy nested relational database model and algebra. *IFIP International Federation for Information Processing*, 221, 285-288.
- Majumdar, A. K., Bhattacharya, I., & Saha, A. K. (2002). An object-oriented fuzzy data model for similarity detection in image databases. *IEEE Transactions on Knowledge and Data Engineering*, 14(5), 1186-1189.
- Marín, N., Medina, J. M., Pons, O., Sánchez, D., & Vila, M. A. (2003). Complex object comparison in a fuzzy context. *Information and Software Technology*, 45(7), 431-444.
- Marín, N., Pons, O., & Vila, M. A. (2001). A strategy for adding fuzzy types to an object-oriented database system. *International Journal of Intelligent Systems*, 16(7), 863-880.

- Marín, N., Vila, M. A., & Pons, O. (2000). Fuzzy types: A new concept of type for managing vague structures. *International Journal of Intelligent Systems*, 15, 1061-1085.
- Prade, H., & Testemale, C. (1984). Generalizing database relational algebra for the treatment of incomplete or uncertain information. *Information Sciences*, 34, 115-143.
- Raju, K. V. S. V. N., & Majumdar, A. K. (1988). Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database systems. *ACM Transactions on Database Systems*, 13(2), 129-166.
- Rundensteiner, E. A., Hawkes, L. W., & Bandler, W. (1989). On nearness measures in fuzzy relational data models. *International Journal of Approximate Reasoning*, 3, 267-298.
- Saxena, P. C., & Tyagi, B. K. (1995). Fuzzy functional dependencies and independencies in extended fuzzy relational database models. *Fuzzy Sets and Systems*, 69, 65-89.
- Shenoi, S., & Melton, A. (1989). Proximity relations in the fuzzy relational databases. *Fuzzy Sets and Systems*, 31(3), 285-296.
- Sözat, M. I., & Yazici, A. (2001). A complete axiomatization for fuzzy functional and multivalued dependencies in fuzzy database relations. *Fuzzy Sets and Systems*, 117(2), 161-181.
- Takahashi, Y. (1991). A fuzzy query language for relational databases. *IEEE Transactions on Systems, Man and Cybernetics*, 21(6), 1576-1579.
- Takahashi, Y. (1993). Fuzzy database query languages and their relational completeness theorem. *IEEE Transactions on Knowledge and Data Engineering*, 5(1), 122-125.
- Tripathy, R. C., & Sakena, P. C. (1990). Multivalued dependencies in fuzzy relational databases. *Fuzzy Sets and Systems*, 38(3), 267-279.
- Umano, M., & Fukami, S. (1994). Fuzzy relational algebra for possibility-distribution-fuzzy-relational model of fuzzy data. *Journal of Intelligent Information Systems*, 3, 7-27.
- Van Gyseghem, N. V., & Caluwe, R. D. (1998). Imprecision and uncertainty in UFO database model. *Journal of the American Society for Information Science*, 49(3), 236-252.
- Vila, M. A., Cubero, J. C., Medina, J. M., & Pons, O. (1996). A conceptual approach for deal with imprecision and uncertainty in object-based data models. *International Journal of Intelligent Systems*, 11, 791-806.
- Yang, Q., Zhang, W. N., Liu, C. W., Wu, J., Yu, C. T., Nakajima, H et al. (2001). Efficient Processing of Nested Fuzzy SQL Queries in a Fuzzy Database, *IEEE Transactions on Knowledge and Data Engineering*, 13(6), 884-901.
- Yazici, A., Buckles, B. P., & Petry, F. E. (1999). Handling complex and uncertain information in the ExIFO and NF2 data models. *IEEE Transactions on Fuzzy Systems*, 7(6), 659-676.
- Yazici, A., George, R., & Aksoy, D. (1998). Design and implementation issues in the fuzzy object-oriented data model. *Information Sciences*, 108(1/4), 241-260.
- Yazici, A., Soysal, A., Buckles, B. P., & Petry, F. E. (1999). Uncertainty in a nested relational database model. *Data & Knowledge Engineering*, 30(3), 275-301.
- Yazici, A., & Koyuncu, M. (1997). Fuzzy object-oriented database modeling coupled with fuzzy logic. *Fuzzy Sets and Systems*, 89(1), 1-26.
- Yazici, A., & Sözat, M. I. (1998). The integrity constraints for similarity-based fuzzy relational databases. *International Journal of Intelligent Systems*, 13(7), 641-660.
- Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8(3), 338-353.
- Zadeh, L. A. (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1(1), 3-28.
- Zhang, W. N., & Wang, K. (2000). An efficient evaluation of a fuzzy equi-join using fuzzy equality indicators. *IEEE Transactions on Knowledge and Data Engineering*, 12(2), 225-237.
- Zvieli, A., & Chen, P. P. (1986). Entity-relationship modeling and fuzzy databases. *Proceedings of the 1986 IEEE International Conference on Data Engineering* (pp. 320-327).

## KEY TERMS

**Conceptual Design of Fuzzy Databases:** It refers to the conversion of fuzzy conceptual data models to fuzzy database models. The focus is on developing the rules of mapping fuzzy conceptual data models to fuzzy database models.

**Fuzzy Conceptual Data Models:** Being the extension of traditional conceptual data models, such as ER/EER and UML fuzzy conceptual data models, which can model imperfect data and semantic relationships at a high level of data abstraction.



## **Fuzzy Database Modeling**

**Fuzzy Database Modeling:** In addition to the issues of fuzzy database models, fuzzy conceptual data models, and conceptual design of fuzzy databases, fuzzy database modeling has a focus on fuzzy database systems and discusses (fuzzy) query, (fuzzy) data handling, database theory (e.g., fuzzy data dependencies and formalization in fuzzy relational databases), database implementation, and so forth.

**Fuzzy Database Models:** The database models which have the ability to store and handle fuzzy data, mainly including fuzzy relational database model, fuzzy nested relational database model, and fuzzy object-oriented database model.

**Fuzzy Set:** The set in which elements are associated with membership degrees in  $(0, 1)$  to indicate how they

belong to the set. Fuzzy-set theory was originated by L. A. Zadeh in 1965 and can be used for imprecise information processing.

**Possibility Distribution:** For a fuzzy set, when the membership degree associated with an element is viewed to be a measure of the possibility that a variable has the value of the element, the fuzzy set becomes a possibility distribution.

**Similarity Relation:** Similarity relation is also used to describe fuzzy data, whereby the fuzziness comes from the similarity relations between two values in a universe of discourse, not from the status of an object itself. So similarity relations are used to represent the degree similarity of two values from the same universe of discourse.

**F**



# Generic Model Management

Zinovy Diskin<sup>1</sup>

SWD Factory, Latvia and Universal Information Technology Consulting, USA

Boris Kadish

SWD Factory, Latvia

Generic model management (gMMt) is a novel view on classical and modern metadata management problems. The present article surveys the goals, components, pros and cons of gMMt, and major problems cited in the literature. It argues that some methodology developed in abstract mathematics can be extremely helpful for the field and is capable of providing it with a convenient notation, semantic foundations and truly generic specification patterns. The two other articles, titled *Mathematics of Generic Specifications for Model Management, I* (further referred to as Math-I, see p. 351), and *Mathematics of Generic Specifications for Model Management, II* (further referred to as Math-II, see p. 359), give some evidence to these claims by demonstrating how the machinery works in a series of examples.

## WHY: FROM ELEMENT-AT-A-TIME TO MODEL-AT-A-TIME PROGRAMMING

Many data management routines include *metadata* applications that manipulate descriptions of data, usually called *schemas*, rather than the data itself. Typical examples are database design, schema integration and evolution, reverse engineering, data integration and translation, or data warehousing. Lately, the Web's dramatically rapid invasion into the field has added to this list several new tasks: ontology engineering and integration, Web site design, and XML wrappers generation. It has also multiplied the importance and diversity of versions of classical tasks by a big coefficient of e-commerce applications. In the OO jargon, data schemas (more generally, metadata artifacts) are often called *models*, so applications listed above can be classified as *model management* (MMt).

Along with models, MMt includes specifying and operating relations between models, which are usually called *model mappings* in the literature. Some examples are mappings between ER- or UML-diagrams on consecutive stages of design or between different releases of a database schema; mappings and their inverses between

ER-diagrams and SQL schemas implementing them; mappings between XML schemas to manage message translation; and various sorts of mappings between various UML diagrams either of one sort (*homogeneous*) or between different sorts (*heterogeneous* model transformation). The construct of view, well known in the relational data model, also presents nothing but a special syntax for specifying a mapping between relational schemas. In a sense, MMt is all about mappings.

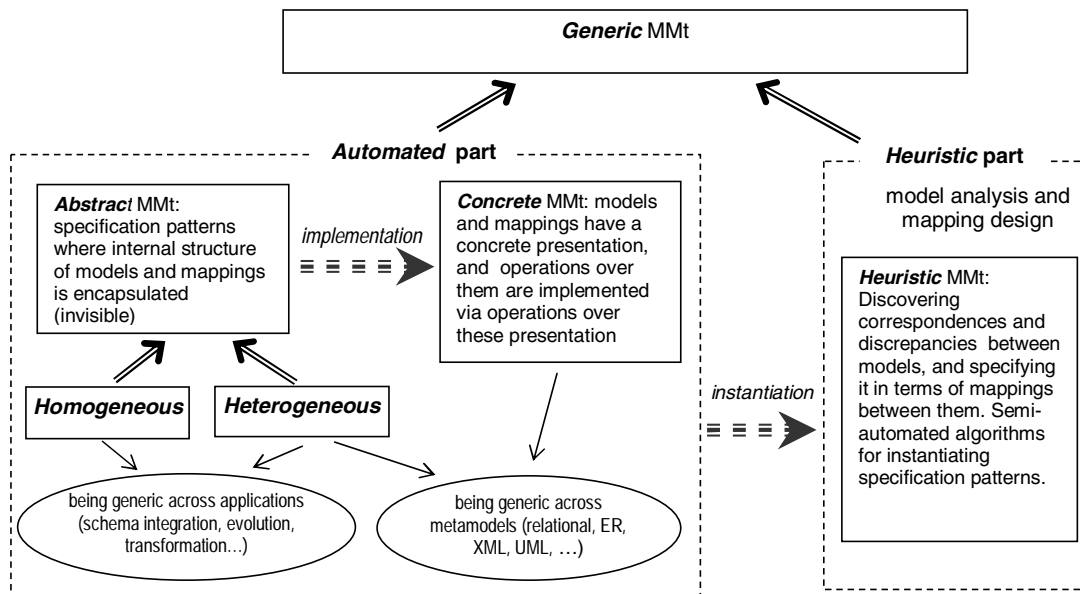
A commonly accepted standard approach to implementing MMt tasks is to present models and mappings as collections of objects and to program manipulations with them via programming manipulations with objects they consist of. Bernstein (2003) calls this *object-at-a-time* programming. A better term might be *element-at-a-time* programming, to emphasize working on the level of elements from which models and mappings are built. Though it does the job, element-wise programming is very laborious and error-prone. In a sense, it is similar to record-at-a-time programming in data processing. As is well known, eliminating the latter in modern DBMSs raised data processing technology on a qualitatively new level in programmers' productivity and semantic transparency.

Similarly, we can expect that a *model/mapping-at-a-time* programming environment, where the application programmer can think of MMt routines in terms of operations over models and mappings as integral entities, could essentially facilitate development and maintenance of metadata applications. To be really useful, such an environment should be *generic*, that is, be applicable to a wide range of MMt tasks involving a wide range of models of different types, i.e., of different *metamodels*. In this way we come to the idea of *generic MMt* (gMMt) environment manifested by Bernstein, Halevy, and Pottinger (2000).

## WHAT: ABSTRACT, CONCRETE AND HEURISTIC PARTS OF GENERIC MMt

To achieve its goals, gMMt must resolve the following three major groups of problems, which are schematically presented in Figure 1.

Figure 1. Three parts of generic MMt



A. **Abstract gMMt—Genericness across applications:**

First of all, we need a generic way of specifying collections of models and mappings. Then we need to find a set of basic operations with models and mappings so that any practically important MMt procedure could be presented as a composition of basic operations. In other words, like we need data definition and manipulation language in data management, in MMt we need *model and mapping definition and manipulation language*. Table 1 compares MMt concepts to be developed with their analogs in the relational view to data management. We call this part of MMt *abstract* since here the internal structure of models and mappings is encapsulated, and they are treated as holistic abstract entities.

Abstract MMt can be divided into two parts, *homogeneous* and *heterogeneous*, dependant on whether the models we deal with are of the same or different types (metamodels). The most important issue in heterogeneous MMt is model translation, and the most difficult

problem in abstract MMt is how to specify it in a generic way (so that, for example, transformations of ER-diagram into an SQL schema and the latter into an XML DTD would be particular instances of the same specification pattern).

B. **Concrete gMMt—Genericness across metamodels:**

To implement abstract MMt patterns and operations, we need to have some concrete representation of models and mappings. Moreover, this representation should be universal with respect to data models (metamodels) so that such diverse models as relational schemas, XML DTDs, or various dialects of ER- and UML-diagrams would all be instances of the same universal format. The problem is evidently far from being easy.

There is a reasonable fear that even if we find such a universal representation *U*, encoding models/mappings of some *particular* metamodel in *U*-terms can be bulky and unwieldy. Then an MMt system’s genericness (as many other sorts of genericness in software products) will be an asset for tool builders rather than for tool users. A counterproposal might be to implement model definition

Table 1. What is to be done in abstract MMt (to be continued in Math-II, Table 2, p. 362)

	Elementary Units	Repository Structure	Elementary Query	A Complete Query Language
Data Management	Value	Set of relations (tables)	Relational operation	Relational algebra (calculus)
Model Management	Model, mapping	??	??	??

and manipulation languages for each metamodel of interest separately rather than strive for a generic-across-metamodels solution. With this approach, however, in addition to a set of particular MMT implementations  $I_1, \dots, I_n$  for each metamodel  $M_1, \dots, M_n$ , we will need to implement an even greater set of model translations  $I_{ij}$ , since many of the MMT tasks are themselves heterogeneous, for example, integration of heterogeneous data sources on the Web or navigating from ER-diagrams to SQL schemas and then to their XML presentations. Developing and maintaining such a system would be really laborious.

Thus, genericness across metamodels appears to be a necessary working property of MMT systems. To realize it, we must find a model representation format that would combine metamodel universality with clear and compact encoding of particular models.

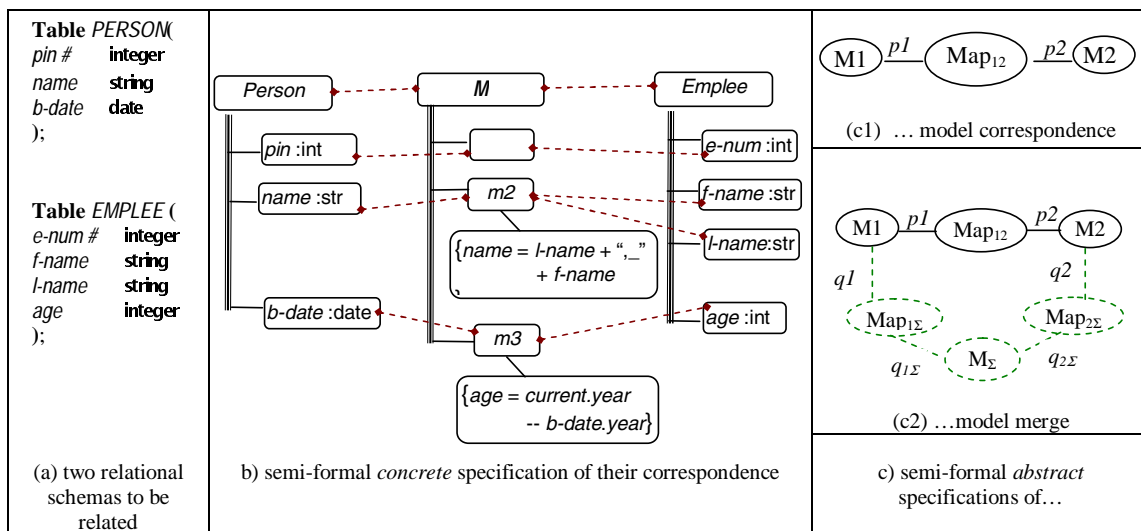
**C. Heuristic gMMT—Discovering and specifying correspondences between models:** As a rule, models involved in an MMT application present different views on the same data universe. Hence, we need to specify overlapping between models and present it as a model mapping. Here we face a highly nontrivial heuristic problem of *discovering* correspondences and discrepancies between models. Indeed, these models were often designed by different people in different contexts and for different goals. They may present the same data in different syntaxes, under different names, and in different formats. Particularly, the infamous problems of synonyms and homonyms in data schemas have to be resolved. In addition, data considered *basic* in one model can be *derived* in another model, that is, to be derivable/computable from the basic data of the other model.

In particular, data in one model can be metadata for another model. All these correspondences must be discovered (the first phase of heuristic MMT) and specified in a suitable format provided by abstract MMT theory (second phase). Of course, the first phase cannot be fully automated. Yet we may hope that some sophisticated algorithms based on special AI techniques (particularly machine learning) could assist the human in the essentially heuristic job of *model matching*.

To illustrate the concepts introduced above, we consider a very simple example in Figure 2. Two simple relational schemas are presented in Column (a) of Figure 2. Suppose we know that table *Person* with primary key column *pin* and table *Emplee* with primary key *e-num* refer to the same real-world objects. In addition, suppose that *Person.name* is nothing but concatenation of *Emplee.l-name* with *Emplee.f-name*. More formally, this description is presented in Column (b), where we use notation of Bernstein (2003) with minor distinctions. In detail, we have introduced a new model, whose each element is a relationship between the original models' elements: Elements *M*, *m1*, and *m3* are 1:1 and *m2* is a 1:2 relationship (of course, general M-N relationships are also possible). In addition, elements *m2* and *m3* have *expressions* attached, which state, correspondingly, that *Person.name* is concatenation of *Emplee.l-name* and *Emplee.f-name*, and *Emplee.age* is equal to the difference between the current year and *Person.bdate.year*.

Abstracting from this concrete specification, we come to the specification in Cell (c1) of Figure 2, where nodes denote models, and edges are relations between them.

Figure 2. Specifying correspondence between models (to be continued in Math-I, Figure 1, p. 352)



Each element of the abstract schema (c1) has an extent shown on the concrete schema (b): Extents of nodes are trees of elements, and extents of edges are sets of pairs (shown by dashed lines) of the corresponding elements.

Finally, Figure 2's Cell (c2) shows an abstract specification of a *model merge* operation: It takes the model diagram (c1) as its input and produces three more nodes with associated four edges:

$$(M_{\Sigma}, Map_{1\Sigma}, Map_{2\Sigma}) = \mathbf{merge}(M1, M2, Map_{12})$$

The concrete “extent” of this operation is a procedure that takes the configuration in Column (b) as its input, manipulates with its elements by predefined rules, and produces a new merge model together with mappings to original models (this is not shown in Column (b)). We will return to this example in *Math-I* and consider it in more details and a better notation.

In this simple example, stating correspondence between models was an easy task. In real situations it may be a far more complicated problem. Finding systematic solutions to such sorts of problems is the essence of heuristic gMMt. Particularly, the heuristic operation **match** takes two models as input and produces the model diagram on Figure 2(c1). Then it becomes the input for **merge**.

All three parts of gMMt are equally important in making it a working environment capable of running practical applications. It is worth stressing, however, that abstract MMt is the specificational cornerstone of the entire building. It provides patterns that, on one hand, must be implemented in a concrete (yet generic) representation and, on the other hand, should be instantiated by data obtained from heuristic algorithms. Compact and adequate abstract MMt patterns would greatly ease the job of the other two MMt components.

## A BIT OF HISTORY AND THE CURRENT SITUATION

MMt problems probably appeared on the stage simultaneously with the first DBMSs. As the technology matured, their weight in the problem basket of data management has been gradually increasing until the recent Web invasion into the field made data integration and transformation—a key MMt problem—a key issue for the entire field.

In each particular area of MMt problems, the database community accumulated a certain experience in both theory and practice. However, inventing a modeling language or writing an MMt application (though laborious) turned out to be much easier heuristically than extracting a common core of particular cases and

abstracting it into a generic framework spanning a wide class of languages and applications. Moreover, the very formulation of desired genericness in precise terms turned out to be a problem because of the absence of language to discuss and specify generic notions (cf. Drew, King, McLeod, Rusinkiewicz, & Silberschatz, 1993).

Nevertheless, under the pressure of practical needs and with a noticeable similarity amongst methods and tools addressing different MMt tasks, particular fragments of the main gMMt ideas have been gradually emerging and developing. Of primary importance among them was the regained interest in mappings between models. The latter first appeared in the literature probably in Paolini and Pelagatti (1977) and Kalinichenko (1978) and then were forgotten until the mid-'90s, when they reappeared on the stage in Diskin and Kadish (1997) and Diskin (1998). These papers proposed an integral formal framework (including a generic notion of query language) for what they called “data-model-independent” metamodeling, and the framework was essentially based on mappings (arrows) between models. Diskin (1999) presented an even more abstract version of the arrow framework and demonstrated its usefulness for clarifying meta-metamodeling issues and formalizing relations between modeling languages, particularly, sublanguages constituting UML. Independently, a detailed study of mappings in the context of the relational data model was undertaken in Miller, Haas, and Hernandez (2000) and Yan, Miller, Haas, and Fagin (2001).

The crucial step towards the formation of gMMt as an integral discipline was made by Bernstein et al. (2000), who manifested the issue and described a detailed agenda of what needed to be done. They once again stated the primary role of model mappings for generic MMt, introduced a set of basic operations with models and mappings, and demonstrated their use in a few application scenarios. Also, they proposed an object tree structure as a generic presentation of models. In fact, Bernstein et al. outlined all three parts of gMMt and initiated a few lines of research in the field. Finally, they coined the very term “generic MMt,” thus establishing a new discipline.<sup>2</sup>

To date, the following results have been achieved. An impressive progress has been made in heuristic MMt, where a few sophisticated algorithms for model matching were elaborated and a few methods developed; see Rahm and Bernstein (2001) for a survey and Halevy and Madhavan (2003) for a discussion of promising novel ideas in the field. A few tools based on semi-automated methods were implemented and tested in practical industrial projects (Madhavan, Bernstein, & Rahm, 2001; Mork & Bernstein, 2004; Velegarakis, Miller, & Popa, 2003; Yan et al., 2001). There has been a great progress in



Table 2. Problems of generic MMt by Bernstein (2003)

<b>Machineries are to be built for:</b>		
<b>1.</b> Filling the gap between models and mappings, which are syntactic structures, and their semantics, where models serve as templates for data instances and mappings govern translation of instances.	<b>2.</b> Representing models in a maximally generic yet tractable way.	<b>3.</b> A general-purpose interface between MMt operators and expression manipulation / inference engines (see Figure 2 for an example of expressions).
<b>Clear and well-defined semantics is needed for:</b>		
<b>4.</b> Operations of (a) Mapping Composition; (b) Model Merge; (c) Model Translation.	<b>5.</b> <i>Generic MMt</i> as a whole. Particularly, evaluation of <i>completeness</i> of a given set of basic operators is required. So far, the very notion of completeness of abstract MMt operations is not developed.	

concrete MMt as well. A convenient graph-based generic presentation for models was developed, and a first full implementation of a gMMt environment was built on its base (Melnik, Rahm, & Bernstein, 2003).

As for abstract MMt, a set of basic operations was further elaborated and applied to a few MMt application scenarios (Bernstein, 2003; Bernstein & Rahm, 2000). Analysis of mappings strongly modulated by the relational data model was continued in Madhavan, Bernstein, Domingos, and Halevy (2002); Velegrakis et al. (2003); and Fagin, Kolaitis, Miller, and Popa (2003). Also, a general categorical framework for MMt, focused on a generic pattern for specifying constraints in data models, was proposed in Alagic and Bernstein (2001).<sup>3</sup> However, despite a few partial advances, abstract MMt remains the least developed part in gMMt. Due to its fundamental role for the entire program, unsolved abstract MMt problems essentially hinder gMMt's progress.

### Problems to be Solved...

Bernstein (2003)—the most complete-to-date presentation of gMMt's goals and means—considers the problems presented in Table 2 to be the most pressing and evaluates the agenda of their solution as requiring “many years and many research groups to develop” (p. 220).

Such a cautious estimation is not surprising: The desired genericness of specifications on one hand and their graph-based (rather than string-based) nature on the other hand lead to an entirely new sort of specification problem. Indeed, so far major theoretical achievements in database research were related to one or another data model, and there are well-developed notions of query, constraint, and instance specialized to a particular data model; e.g., relational (rigorously formalized) or a particular dialect of ER or OO (much less clearly formalized though). Abstract generalization of these notions in a rigorous way presents a new sort of problem, where such

well-known and deserved mathematical means as first-order logic or relational algebra cannot help. Diskin, Kadish, and Piessens (1999) discuss the issue in detail.

### ... and Their Solutions: Category Theory vs. Generic MMt

Fortunately, software engineers and researchers do not need to develop specification foundations for gMMt from scratch. Mathematical *category theory* (CT) is a discipline specially devoted to specifying operations with complex objects in an abstract generic way independent of the objects' internal structure.

Categorically, everything that one wants to say about objects, one must say in terms of *morphisms* (or *arrows* or *mappings*) between objects. Such a reformulation is not evident and is sometimes really nontrivial, but the essential benefit is that the objects' internal structure does not occur in specifications. This allows CT to design really generic patterns (necessarily graph-based) for specifying and manipulating complex structures, and these patterns can be readily employed in gMMt.

There are three major categorical prescriptions that jointly form a framework where each of the problems in Table 2 can be successfully managed. Moreover, this framework eliminates the three major obstacles to realizing a truly generic MMt environment, which so far have been considered insurmountable (Bernstein, 2003):

1. Operation of model translation is inevitably metamodel specific.
2. Mechanisms for dealing with expressions also must be metamodel specific.
3. A compromise between expressiveness of model representation and tractability of operations over it is inevitable.

In this section, we will only name the three concepts



—Kleisly morphism, fibration and sketch—leaving their definitions and explanations of how they work for *Math-I* and *Math-II*.

**Prescription 1 (for homogeneous abstract MMt)**—**Consider model mappings as Kleisly morphisms:** This prescription sets up a proper framework for working with derived data and managing Problems 3 and 4(a, b) and a good part of Problem 5. In fact, Problems 3 and 4(a) are eliminated by reducing them to composition of Kleisly morphisms: Since Kleisly morphisms are functional mappings, their composition is easy and presents a form of term substitution. Kleisly morphisms put MMt on truly arrow (hence, generic) foundations and serve as a main vehicle of MMt’s categorization (see *Math-I* for details). Particularly, their convenience for specifying schema repositories, including relations between schemas (such as views and refinement) and operations over them (like schema integration), was shown in Kadish and Diskin (1996), Diskin and Kadish (1997), and Diskin (1998).

**Prescription 2 (for heterogeneous abstract MMt)**—**In a heterogeneous environment, consider data as a fibration of the data instances over the universe of schemas, in its turn fibred over the universe of metaschemas:** To realize this prescription, we need to choose some generic representation for data and schemas, for example, graphs. Then the fibrational view on data prescribes to supply (label) each item (an object identifier or a reference) in the data graph  $D$  with its type and organize the latter into a graph—the schema of the data  $S$ .<sup>4</sup> Then, data can be seen as a graph mapping  $\delta: D \rightarrow S$ . Being applied to the metalevel, the fibrational view suggests that in a heterogeneous universe, we need to keep each model with its metamodel and label each item of the model with its type from the metamodel. This way we come to a mapping  $\sigma: S \rightarrow M$  with  $M$  a graph specifying the metamodel. The fibrational view on data and metadata allows managing Problems 1 and 4(c) and, jointly with Prescription 1, Problem 5.

Following Prescriptions 1 and 2 we can build a specification framework where any reasonable MMt routine, including data and schema translation, integration, and evolution, is presented as a composition of a few basic diagram operations with data, schemas, metaschemas, and mappings between them. Definitions of these operations do not anyhow depend on the internal structure of data, schemas, and mappings. The basics of the approach are presented in *Math-I* and *II*.

**Prescription 3 (for concrete MMt)**—**Consider models as (visual presentations of categorical) sketches and data as sketch instances in the category of sets and mappings<sup>5</sup>:** It is mathematically proven that any data that can be specified formally can be specified by a sketch as well. Hence, the sketch format gives a really universal specification language of absolute expressiveness. Gen-

erally speaking, the sketch presentation of models is not a must for gMMt, but it greatly simplifies many issues by providing specifications combining the following advantages:

- **Extreme compactness:** A sketch consists of only three sorts of items—nodes, arrows, and diagram predicate declarations.
- **Clear and simple semantics:** For data modeling, nodes are sets, and arrows are mappings between them.
- **Clear and simple formal definitions of model mapping:** This property is especially valid for MMt.

Being applied to the metalevel, where models are data whose schemas are metamodels, Prescription 3 suggests to consider models as instances of sketches specifying metamodels. It allows us to apply many results of categorical logic to MMt problems. Details about sketches and their applications to data modeling can be found in Diskin (2003); Diskin and Kadish (2003); and Diskin, Kadish, Piessens, and Johnson (2000).

## CONCLUSION

Generic MMt is an actively developing novel research field. Its creation has been motivated by purely practical reasons of effective metadata management. Nevertheless, gMMt quickly came to specification issues requiring fairly abstract mathematical means. Fortunately, the latter are already developed in category theory and can be readily adapted for MMt needs.

After a specification framework for gMMt is developed, the following two major tasks are in order. The first is to further develop a theory and algorithms for schema matching and to test them in various practical situations. It is a large and practically unlimited research and industrial field in-between DB and AI. The second major task is to build effective implementations of the gMMt environment and to estimate their efficiency in a variety of contexts and practical applications. Together these tasks form a broad experimental agenda still in its infancy. However, most likely it will rapidly mature and progress under the pressure of such important applications as data warehousing and the Web.

## REFERENCES

Alagic, S., & Bernstein, P. (2001). A model theory for generic schema management. In *Eighth International Workshop on Databases and Programming Languages*,

DBPL'2001 (pp. 228-246).

Bernstein, P. (2003). Applying model management to classical metadata problems. In *International Conference on Innovative Database Research, CIDR'2003* (pp. 209-220).

Bernstein, P., Halevy, A., & Pottinger, R. (2000). A vision for management of complex models. *SIGMOD Record*, 29(4), 55-63.

Bernstein, P., & Rahm, E. (2000). Data warehouse scenarios for model management. In *International Conference on Entity-Relationship Modeling, ER'2000* (pp. 1-15).

Cadish, B., & Diskin, Z. (1996). Heterogeneous view integration via sketches and equations. In *Lecture notes in artificial intelligence: Vol. 1079. Foundations of Intelligent Systems: Ninth International Symposium, ISMIS'96* (pp. 603-612). Springer.

Diskin, Z. (1998). The arrow logic of meta-specifications: A formalized graph-based framework for structuring schema repositories (Tech. Rep. No. TUM-19820). In H. Kilov, B. Rumpe, & I. Simmonds (Eds.), *Seventh OOPSLA Workshop on Behavioral Semantics of OO Business and System Specifications*. Technische Universitaet Muenchen.

Diskin, Z. (1999). Abstract metamodeling, I: How to reason about meta- and metamodeling in a formal way. In K. Baclawski, H. Kilov, A. Thalassinidis, & K. Tyson (Eds.), *Eighth OOPSLA Workshop on Behavioral Semantics* (pp. 32-48) Northeastern University.

Diskin, Z. (2003). Mathematics of UML: Making the odysseys of UML less dramatic. In K. Baclawski, & H. Kilov (Eds.), *Practical foundations of business system specifications* (pp. 145-178). Kluwer.

Diskin, Z., & Kadish, B. (1997). A graphical yet formalized framework for specifying view systems. In *Advances in Databases and Information Systems, ADBIS'97: Vol. 1. Regular papers* (pp. 123-132)., St. Petersburg, Russia: Nevsky Dialect.

Diskin, Z., & Kadish, B. (2003). Variable set semantics for keyed generalized sketches: Formal semantics for object identity and abstract syntax for conceptual modeling. *Data & Knowledge Engineering*, 47, 1-59.

Diskin, Z., Kadish, B., & Piessens, F. (1999). What vs. how of visual modeling: The arrow logic of graphic notations. In H. Kilov, B. Rumpe, & I. Simmonds (Eds.), *Behavioral specifications in businesses and systems* (pp. 27-44). Kluwer.

Diskin, Z., Kadish, B., Piessens, F., & Johnson, M. (2000). Universal arrow foundations for visual modeling. In *Lecture notes in artificial intelligence: Vol. 1889. International Conference on the Theory and Applications of Diagrams* (pp. 345-360). Springer.

Drew, P., King, R., McLeod, D., Rusinkiewicz, M., & Silberschatz, A. (1993). Report on the workshop on semantic heterogeneity and interoperation in multidatabase systems. *SIGMOD Record*, 22(3).

Fagin, R., Kolaitis, P., Miller, R., & Popa, L. (2003). Data exchange: Semantics and query answering. In *International Conference on Database Theory, ICDT'2003* (pp. 207-224).

Goguen, J., & Burstall, R. (1992). Institutions: Abstract model theory for specification and programming. *Journal of ACM*, 39(1), 95-146.

Halevy, A., & Madhavan, J. (2003). Corpus-based knowledge representation. In *International Joint Conference on Artificial Intelligence, IJCAI'03*.

Kalinichenko, L. (1978). Data model transformation method based on axiomatic data model extension. In *International Conference on Very Large Data Bases, VLDB'78* (pp. 549-555).

Madhavan, J., Bernstein, P., Domingos, P., & Halevy, A. (2002). Representing and reasoning about mappings between domain models. In *18th Conference of American Association for Artificial Intelligence, AAAI'2002*.

Madhavan, J., Bernstein, P., & Rahm, E. (2001). Generic schema matching with Cupid. In *International Conference on Very Large Databases, VLDB'2001*.

Melnik, S., Rahm, E., & Bernstein, P. (2003). Developing metadata-intensive applications with Rondo. *Journal of Web Semantics*, 1, 47-74.

Miller, R., Haas, L., & Hernandez, M. (2000). Schema mapping as query discovery. In *International Conference On Very Large Databases, VLDB'2000*.

Mork, P., & Bernstein, P. (2004). Adapting a generic match algorithm to align ontologies of human anatomy. In *International Conference on Data Engineering, ICDE'2004* (pp. 787-790).

Paolini, P., & Pelagatti, G. (1977). Formal definition of mapping in a database. In *ACM SIGMOD Conference on Management of Data* (pp. 40-46).

Rahm, E., & Bernstein, P. (2001). A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4), 334-350.

Velegrakis, Y., Miller, R., & Popa, L. (2003). Mapping adaptation under evolving schemas. In *International Conference on Very Large Databases, VLDB'2003*.

Yan, L.-L., Miller, R., Haas, L., & Fagin, R. (2001). Data-driven understanding and refinement of schema mappings. In *ACM SIGMOD Conference on Management of Data*.

## KEY TERMS

**Data Model:** A specification language where we can model and talk about models of a given class and their instances, manipulations with instances and models, queries, and constraints. A typical example of a well-developed data model is the relational model. ER data model, though less formal, is another example.

**Data Transformation/Translation:** Transformation of data stored in one format (metamodel) to another format.

**Generic Model Management (gMMt):** An MMt environment/system applicable to a wide range of MMt tasks across a wide range of metamodels.

**Instance (of a Model):** A set of data items structured according to the model. Data *instantiate* the model (or data schema in this context).

**Metadata:** Data specifying data, their structure, and presentation.

**Metamodel (of a Given Class of Models):** A model whose instances are the models of the class. Models of a given metamodel are called *similar*.

**Model:** A metadata artifact such as relational or XML schema, interface definition, or Web site layout.

**Model Mapping or Morphism:** Informally, a correspondence between models. Formally, it is a function (arrow) sending elements of the source model to elements of the target model. The graph of this function can be reified as a special model, often also called *model mapping* in the literature. We prefer the term *correspondence model* for reified mappings and use *model mapping* in the narrow sense of mappings-as-functions.

## ENDNOTES

<sup>1</sup> Partially supported by Grant 05.1526 from the Latvian Council of Science.

<sup>2</sup> Inventing a good term is not just linguistic sugar; rather, it is an act of stating a notion or concept, and it may significantly speed up the progress in the field.

<sup>3</sup> The framework is based on the notion of *institution* (Goguen & Burstall, 1992), which was invented for “model management” in mathematical logic (where models are logical theories). A discussion of institutions’ applicability to MMt problems can be found in Diskin (1999), where a bit more general version of institutions was developed for precise formulation of meta-metamodeling concepts.

<sup>4</sup> This is nothing but the XML view on data, and indeed an XML document can be viewed as a fibration of data over tags, as soon as we arrange data and the set of tags into graphs. In a sense, XML’s authors invented fibrations independently of category theorists.

<sup>5</sup> Sketch is a graph-based format used in category theory for specifying mathematical structures (see *Math-I* for details).

# Geometric Quality in Geographic Information

**José Francisco Zelasco**

*UNBA, Argentina and UNCPBA, Argentina*

**Gaspar Porta**

*Carthage College, USA*

**José Luis Fernandez Ausinaga**

*Pop-Vision Company, Argentina*

## INTRODUCTION

A typical way to build surface numerical models or Digital Elevation Models (DEMs) for Geographical Information Systems (GIS) is by processing the stereo images obtained from, for example, aerial photography or SPOT satellite data. These GIS can perform many computations involving their geographic databases. The quality control of a geographic database, and in particular the topological and geometric integrity, are, therefore, important topics (Guptill & Morrison, 1995; Harvey, 1997; Laurini & Milleret-Raffort, 1993; Ubeda & Servigne, 1996). The geometric quality control of the stored DEM is what we are concerned with here. "Quality" means the geometric precision measured in terms of the difference between a DEM and a reference DEM (R-DEM). We assume the R-DEM is a faithful model of the actual surface. Its point density may be greater than the DEM point density.

## BACKGROUND

In the literature, several unsatisfactory solutions were proposed for the DEM control with respect to a reference. A critical problem in the error estimation (evaluated using the difference referred to in the previous paragraph) is to establish for each selected point of the DEM the corresponding homologous point in the R-DEM. Other kinds of problems and errors are related to the existence of aberrant points, systematization, and so forth. These problems were studied for horizontal errors in maps in Abbas (1994), Grussenmeyer, Hottier, and Abbas (1994), and Hottier, (1996a). These authors found that the dissymmetry model-reference was the most important factor to determine homologous pairs.

Several solutions have been proposed for the punctual control method (e.g., recognition algorithms, filtering methods, adjustment of histograms to theoretical laws) without obtaining completely satisfactory results (Dunn, Harrison, & White, 1990; Lester & Chrisman, 1991). Later,

Abbas (1994), Grussenmeyer (1994), and Hottier (1996a) presented an alternative to the punctual control method: the linear control method based on the dissymmetry of the Hausdorff distance. Habib (1997) analyzed precision and accuracy in altimetry and mentioned some of the proposals of the last decade for the elevation control of quality.

In the case of the DEMs, to assess the difference that gives rise to the error we wish to compute, we need to identify without ambiguity each point  $M$  in the DEM with its homologous point  $P$  in the R-DEM. Two reasons that make this task difficult follow:

1. Many points  $M$ , such as those situated on regular sides, which are indistinguishable from their neighbors, are not identifiable. Potentially identifiable points are those located on sharp slope variations and possibly those with zero slope (tops, bottoms, and passes).
2. A point identifiable on the DEM is not necessarily identifiable in the R-DEM because of a difference in scale (generalization) or aberrant errors.

Finding identifiable homologous pairs of points is difficult for an operator, and automating this is a very delicate process.

In the case of the precision assessment of a DEM from an ASTER (Advanced Spaceborne Thermal Emission and Reflection Radiometer; Hirano, Welch, & Lang, 2002), profiles or benchmarks (particular points) are used for the elevation accuracy assessment. This elevation accuracy assessment refers to the vertical error of the DEM. However, the horizontal error is not taken into account when using profiles and, also, there is a model-reference discrepancy and the choice and number of the benchmarks might not be a good stochastic sample of points.

In light of these difficulties, Zelasco and Ennis (2000) proposed an estimator for the variances of the vertical and horizontal errors. The values obtained for the estimator according to the type of terrain, its unevenness, and the number of sample points in the DEM were studied using



simulations (Zelasco, 2002; Zelasco, Ennis, & Blangino, 2001). Throughout these studies, it is assumed that the errors in elevation and in the horizontal plane are independent normal random variables—in agreement with Liapounov’s theorem (Hottier, 1996b). This proposed method is called the Perpendicular Distance Estimation Method (PDEM).

Let  $e(M_k) = M_k - P_k$  where  $M_k, P_k$  are the  $k$ -selected homologous pairs of points in the DEM and R-DEM, respectively. Estimating  $\sigma^2(e)$  (variance of the error as distance between  $M$  and  $P$ ) without measuring each vector  $e(M_k)$  completely, is the key advantage of the proposed PDEM. Only the projection of each  $e(M_k)$  in particular directions matters. Given an  $M_k$  it becomes unnecessary to find the homologous  $P_k$  in the R-DEM. How this is done is the subject of the following section.

For three-dimensional DEMs built with the usual techniques, such as photogrammetry or remote sensing, there are expected normal deviations in altimetry ( $\sigma_z$ ) and in the horizontal plane ( $\sigma_p$ ) that are a priori known. They are generally different from each other due to the manner in which the values of the  $z$  coordinate (altimetry) and the  $x, y$  coordinates (the horizontal plane) are evaluated. The goal of this PDEM is to evaluate the actual errors (a posteriori) of a given DEM.

## THE QUALITY EVALUATION METHOD

### Description of the PDEM

The gap between the DEM and the R-DEM is represented by a function that links each point of the DEM to the vector  $e(M) = M - P$  in  $R^3$  where  $P$  is the homologous point to  $M$  in the R-DEM.  $M$  and  $P$  represent the same relief element; however,  $P$  is not vertically aligned with  $M$ . The DEM precision is a function of the vectors  $e(M_k)$  of a sample of  $\{M_k\}$  points. This function is used in the construction of the covariance matrix  $\sigma^2(e)$ . If the errors in the three directions are different, for instance, in interferometry radar, the rank of  $\sigma^2(e)$  is three. Its eigenvalues are the variances in the three main coordinates.

When the errors in the orthogonal directions in the plane are equal (stereo images techniques), we can consider that the rank of the covariance matrix  $\sigma^2(e)$  is two. The eigenvalues are the variances  $\sigma_z^2$  and  $\sigma_x^2 = \sigma_y^2 = \sigma_{x,y}^2$ .

In a rotated frame of reference, the new covariance matrix will no longer be diagonal. The new components are expressed in terms of the eigenvalues of  $\sigma^2$ . The following expression corresponds to the first component  $\sigma_x'^2$  of the new covariance matrix:

$$\sigma_x'^2 = a_{11}^2 \sigma_x^2 + a_{12}^2 \sigma_y^2 + a_{13}^2 \sigma_z^2 \quad (1)$$

where  $a_{ij}$  are the managing cosines of the unit vector in the  $x'$  coordinate of the rotated reference frame.

The value of the variance in the  $x'$  direction is equal to the square of the distance between the two planes normal to the  $x'$  direction: one tangent to the distribution indicative ellipsoid (whose parameters are  $a = \sigma_x$ ;  $b = \sigma_y$ ;  $c = \sigma_z$ ) and the other passing through its center.

In Figure 1, the plane normal to the  $x'$  direction and tangent to the distribution indicative ellipsoid is drawn in the general case.

The algebraic form of equation (1) is used to obtain the variance in any direction  $d$ , replacing the coefficients  $a_{ij}$  with the managing cosines ( $\cos \alpha, \cos \beta, \cos \gamma$ ) corresponding to this direction  $d$ :

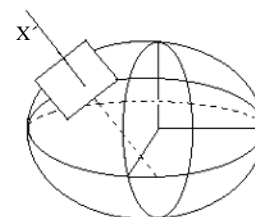
$$\sigma_d^2 = \sigma_x^2 \cos^2 \alpha + \sigma_y^2 \cos^2 \beta + \sigma_z^2 \cos^2 \gamma \quad (2)$$

$\sigma_d^2$  can be estimated with  $\sum e_i^2/n$   $0 < i < I$  where the  $e_i$  are the error components in the  $d$  direction.

We construct a mesh of triangles from the points of the R-DEM. We call this a model of the surface. Given a point  $M$  in the DEM, its projection onto the  $x, y$  plane is inside the projection of a unique triangle  $T$  from this model. We call this the corresponding triangle  $T$  to  $M$ . The  $e_i$  component in the direction  $d_i$  is the distance  $d_i$  from each point to the plane of the corresponding triangle. The squares of the distances  $d_i$ , between each point of the model and the plane of the corresponding triangle, allow us to establish a least squares estimator for the assessment of  $(\sigma_x^2, \sigma_y^2, \sigma_z^2)$ .

Recall that if the errors in the horizontal plane are independent of the direction, the problem is reduced from three to two unknowns (see Figure 2). The  $\sigma_x = \sigma_y$  and the distribution indicative ellipsoid is a revolution ellipsoid. The maximum slope direction in each R-DEM triangle (equal to the angle made by the normal to the triangle and the  $z$  coordinate) will determine the managing cosines direction in two dimensions. The position of the homologous point in the R-DEM does not need to be known. This is the strength of the PDEM. By having determined the corresponding triangle, we capture the required information without needing to find the particular homologous

Figure 1. Distribution indicative ellipsoid and tangent plane  $\Pi$





point of the R-DEM. Therefore, it is only needed to determine the distance  $d_i$ , which is the error component in the direction normal to the triangle.

From (2) and with

$$\sigma_x^2 = \sigma_y^2 = \sigma_{x,y}^2 \quad (3)$$

we get

$$\sigma_d^2 = \sigma_{x,y}^2(\cos^2\alpha + \cos^2\beta) + \sigma_z^2\cos^2\gamma \quad (4)$$

and finally

$$\sigma_d^2 = \sigma_{x,y}^2\sin^2\gamma + \sigma_z^2\cos^2\gamma \quad (5)$$

If we assume that we have only one value of the distance  $d_i$  for each direction  $d_i$ , we may consider that  $d_i^2$  is an estimator of  $\sigma_d^2$ , so we get the solution with the  $i$  relations

$$d_i^2 = \sigma_{x,y}^2\sin^2\gamma + \sigma_z^2\cos^2\gamma, \quad (6)$$

which is more precise than using  $\sigma_d^2$  estimated by means  $\sum e_i^2/n$   $0 < i < I$ . Additionally this avoids having to give a different weight to each equation.

## HOW TO EMPLOY THE PDEM

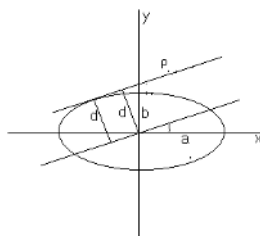
Recall that an R-DEM will be used as the control for a DEM, real or simulated. Also recall that an R-DEM, being a discrete set, allows constructing a mesh of  $K$  triangles defined by the points on the reference surface. Each triangle  $T_k$ , ( $k = 1, \dots, K$ ), belongs to a plane which has a normal unitary vector  $n_k$ , ( $k = 1, \dots, K$ ).

The mesh of  $K$  triangles of the R-DEM is constructed using, for instance, the Delaunay method in two dimensions.

In the case of a simulation, we have to

- determine the mass center of each triangle.
- adopt a standard deviation for each coordinate  $\sigma_x$ ,  $\sigma_y$ , and  $\sigma_z$  and add three random normal variables to the

Figure 2. Distribution indicative ellipse and tangent straight line  $\rho$



center mass coordinates to simulate the points of the DEM to be evaluated. With  $\sigma_x = \sigma_y$  we evaluate  $\sigma_{x,y}$  (horizontal plane standard deviation). In this case, the distribution indicative ellipsoid is a revolution surface.

In the case of a real DEM, we have to

- determine the triangle of the R-DEM containing the homologous point of each point of the DEM. Recall that the procedure is as follows: Given a DEM point  $M_s$ , its homologous point  $P$  in the R-DEM will be in a plane define by the triangle  $T_s$ , if the projection of the point  $M_s$  on the  $x,y$  plane is inside the projection of the triangle  $T_s$  on the same plane. We establish a practical limit to avoid border problems given by the a priori standard deviation  $\sigma_p$  (in the horizontal plane), which allows us to select the points of the DEM in the sample. If the horizontal distance between the projection of  $M_s$  and the limits of the projected triangle is at least three times greater than  $\sigma_p$ , the point is accepted. Otherwise, it is rejected. This is because  $\sigma_p$  is the standard deviation of a normal random variable.

In both cases, we have to

- calculate the distance from each point (point of the simulated or of the real DEM) to the plane of the corresponding triangle.
- evaluate, using the PDEM and the standard deviation of the DEM in altimetry ( $\sigma_z$ ) and in the horizontal plane ( $\sigma_{x,y}$ ). In the case of the simulation, their values should be similar to the ones already used.

Note 1: Knowledge of the  $\gamma$  (in (5)) angle makes it possible to select an R-DEM with more or less unevenness.

Note 2: For each one of the planes having the direction  $d$  given by the normal unitary vector  $n_k$  in question, the distance from the point  $M_j$ , ( $j = 1 \dots J < = I$ ) to the corresponding plane is a measure (in the direction  $d$ ) of the error  $e_d$

## DISCUSSION

Evaluating the horizontal error is the most delicate task in the geometric quality control of a geographical database. When the unevenness of the R-DEM surface is insufficient, there is not enough information to evaluate the horizontal error; the assessment of the horizontal error will not be reasonably precise. With enough unevenness, the horizontal error can be comfortably esti-

mated. The elevation error estimation is better than the horizontal one. The quality of the horizontal error assessment will depend on the unevenness of the reference surface as well as on the dimension of the sample (Zelasco, 2002).

With simulations, the error precision values can be calculated, but with a real DEM, the error precision is unknown (it can only be estimated). Applying the PDEM, two values are obtained: the elevation error and the horizontal error of a DEM. With these estimated values (in elevation and in the horizontal plane) simulations may be performed to evaluate the standard deviation of the estimated errors. These standard deviations determine whether to keep or to reject the estimated errors—particularly in the case of the horizontal error.

## FUTURE TRENDS

It would be quite interesting to consider the effects of different error values for the  $x$  and  $y$  coordinates. It might also be of interest to study how to modify the proposed method, taking into account the correlation between the error in elevation and the error in a particular direction of the horizontal plane. In this way, the PDEM may be extended to the case of the Interferometry SAR.

## CONCLUSION

Estimating the elevation error ( $\sigma_z$ ) by means of the commonly used vertical distance estimator gives poor results with an uneven surface. Using the PDEM, the elevation error evaluation will almost always be very good. The horizontal error evaluation ( $\sigma_{x,y}$ ) will be satisfactory when the unevenness and the number of points of the sample are appropriate.

## REFERENCES

Abbas, I. (1994). *Base de données vectorielles et erreur cartographique: Problèmes posés par le contrôle ponctuel; Une méthode alternative fondée sur la distance de Hausdorff: Le contrôle linéaire*. Thèse, Université Paris 7.

Abbas, I., Grussenmeyer P., & Hottier, P. (1994). *Base de Données géolocalisées: Estimation de l'erreur cartographique: Une méthode fondée sur la distance Hausdorff: le contrôle linéaire. Remarques sur le contrôle ponctuel*. Notes de cours, ENSG IGN, France.

Dunn, R., Harrison, A. R., & White, J. C. (1990). Positional accuracy and measurement error in digital databases of land use: An empirical study. *International Journal of Geographic Information Systems*, 4(4), 385-398.

Grussenmeyer, P., Hottier P., & Abbas, I. (1994). Le contrôle topographique d'une carte ou d'une base de données constituées par voie photogrammétrique. *Journal XYZ 59, Association Française de Topographie*, 39-45.

Guptill, S. C., & Morrison, J. L. (Eds.). (1995). *Elements of spatial data quality*. Oxford, UK: Elsevier.

Habib, M. (1997). *Etude, par simulation, de la précision altimétrique et planimétrique d'un MNT obtenu par corrélation d'images spatiales (capteur à balayage)—Précision d'un MNT régulier—Erreur de rendu d'un semis régulier—Description d'une structure de MNT régulier*. Thèse, Université Paris 7.

Harvey, F. (1997). Quality needs more than standards. In M. Goodchild & R. Jeansoulin (Eds.), *Data quality in geographic information: From error to uncertainty* (pp. 37-42). Paris: Hermès.

Hirano A., Welch R., & Lang, H. (2002). Mapping from ASTER stereo image data: DEM validation and accuracy assessment. *Photogrammetry and Remote Sensing*, 57, 356-370.

Hottier, P. (1996a). *La méthode du contrôle linéaire en planimétrie - Propositions pour une typologie des détails anormaux*. Rapport de recherche, ENSG, IGN, France.

Hottier, P. (1996b). *Précis de statistiques*. ENSG, IGN, France.

Hottier, P. (1996c). Qualité géométrique de la planimétrie. Contrôle ponctuel et contrôle linéaire. Dossier: La notion de précision dans le GIS. *Journal Géomètre 6*, juin, Ordre des Géomètres-Experts Français, 34-42.

Lester, M., & Chrisman, N. (1991). Not all slivers are skinny: A comparison of two methods for detecting positional error in categorical maps. *GIS/LIS*, 2, 648-656.

Laurini, R., & Milleret-Raffort, F. (1993). *Les bases de données en géomatique*. Paris: Editions HERMES.

Ubeda, T., & Servigne, S. (1996, September). Geometric and topological consistency of spatial data. *Proceedings of the 1<sup>st</sup> International Conference on Geocomputation*. (Vol. 1, pp. 830-842). Leeds, UK.

Zelasco, J. F. (2002). Contrôle de qualité des modèles numériques des bases de données géographiques. *Journal XYZ 90, Association Française de Topographie*, 50-55.

Zelasco, J. F., Ennis, K., & Blangino, E. (2001). Quality control in altimetry and planimetry for 3D digital models. Paper presented at the 8<sup>th</sup> European Congress for Stereology and Image Analysis, Bordeaux, France.

Zelasco, J. F., & Ennis, K. (2000). *Solución de un problema en la estimación de variancias en un caso especial en el que no se pueden aplicar estimadores habituales*. XXVIII Coloquio Argentino de estadística, Posadas, Misiones, Argentina.

## KEY TERMS

**Covariance Matrix:** The square  $n \times n$  of which the entries are the pairwise correlations of the variables of a random vector of length  $n$ ; the  $(i,j)$ th entry is the correlation between the  $i$ th and the  $j$ th variables.

**DEM Geometric Quality:** Geometric precision measured in terms of the difference between a digital elevation model (DEM) and a reference DEM (R-DEM).

**Digital Elevation Model (DEM):** The set of points in a three-dimensional coordinate system modeling a real object's surface.

**Geographic Information System (GIS):** Tool for processing localized information. A GIS will model and locate the spatial data of a real phenomenon.

**Homologous Points:** The point in the DEM and the point in the R-DEM modeling the same point in the real surface. Their distance is the error of the DEM point, assuming the points in the R-DEM are without error.

**Photogrammetry:** Technique permitting the setup of the coordinates of points (DEM) of an object surface by processing stereo images of the object's surface.

**Revolution Ellipsoid:** Three-dimensional solid obtained from revolving an ellipse about one of its axes.

**Spatial Data:** Information related to a real system involving geometric (dimension, shape) and topological (relative position) aspects of it.

**Stereo Images:** Two or more images of a real object's surface obtained from different points of view.

**Variance Estimation:** Average of the sum of the squares of the differences of the values attained by a random variable from its mean.

# Hierarchical Architecture of Expert Systems for Database Management

**Manjunath R.**

*Bangalore University, India*

## INTRODUCTION

Expert systems have been applied to many areas of research to handle problems effectively. Designing and implementing an expert system is a difficult job, and it usually takes experimentation and experience to achieve high performance. The important feature of an expert system is that it should be easy to modify. They evolve gradually. This evolutionary or incremental development technique has to be noticed as the dominant methodology in the expert-system area.

Knowledge acquisition for expert systems poses many problems. Expert systems depend on a human expert to formulate knowledge in symbolic rules. It is almost impossible for an expert to describe knowledge entirely in the form of rules. An expert system may therefore not be able to diagnose a case that the expert is able to. The question is how to extract experience from a set of examples for the use of expert systems. Machine learning algorithms such as “learning from example” claim that they are able to extract knowledge from experience. Symbolic systems as, for example, ID3 (Quinlan, 1983) and version-space (Mitchell, 1982) are capable of learning from examples. Connectionist systems claim to have advantages over these systems in generalization and in handling noisy and incomplete data. For every data set, the rule-based systems have to find a definite diagnosis. Inconsistent data can force symbolic systems into an indefinite state. In connectionist networks, a distributed representation of concepts is used. The interference of different concepts allows networks to generalize

A network computes for every input the best output. Due to this, connectionist networks perform well in handling noisy and incomplete data. They are also able to make a plausible statement about missing components. A system that uses a rule-based expert system with an integrated connectionist network could benefit from the described advantages of connectionist systems.

## BACKGROUND

Maintenance of databases in medium-size and large-size organizations is quite involved in terms of dynamic

reconfiguration, security, and the changing demands of its applications. Here, compact architecture making use of expert systems is explored to crisply update the database. An architecture with a unique combination of digital signal processing/information theory and database technology is tried. Neuro-fuzzy systems are introduced to learn “if-then-else” rules of expert systems. Kuo, Wu, and Wang (2000) developed a fuzzy neural network with linguistic teaching signals.

The novel feature of the expert system is that it makes use of a large number of previous outputs to generate the present output. Such a system is found to be adaptive and reconfigures fast. The expert system makes use of a learning algorithm based on differential feedback. The differentially fed learning algorithm (Manjunath & Gurumurthy, 2002) is introduced for learning. The learning error is found to be minimal with differential feedback. Here, a portion of the output is fed back to the input to improve the performance. The differential feedback technique is tried at the system level, making the system behave with the same set of learning properties. Thus, control of an expert system controls the entire system

## KNOWLEDGE EXTRACTION FROM DIFFERENTIALLY FED NEURAL NETWORKS

The expert systems are organized in a hierarchical fashion. Each level controls a unique set of databases. Finally, the different expert systems themselves are controlled by a larger expert system. This ensures security of databases and selective permissions to their access, (i.e., some of the data needs to be public and the rest has to be private and protected; a concept borrowed from object-oriented programming). Thus, the master expert system can have access to public information. The neural networks are integrated with a rule-based expert system. The system realizes the automatic acquisition of knowledge out of a set of examples. It enhances the reasoning capabilities of classical expert systems with the ability to generalize and the handle incomplete cases. It uses neural nets with differential feedback

algorithms to extract regularities out of case data. A symbolic-rule generator transforms these regularities into rules governing the expert system. The generated rules and the trained neural nets are embedded into the expert system as knowledge bases. In the system diagnosis phase it is possible to use these knowledge bases together with human experts' knowledge bases in order to diagnose an unknown case. Furthermore, the system is able to diagnose and to complete inconsistent data using the trained neural nets exploiting their ability to generalize.

It is required to describe a possible approach for the optimization of the job scheduling in large distributed systems, based on self-organizing neural networks. This dynamic scheduling system should be seen as adaptive middle-layer software, aware of current available resources and making the scheduling decisions using past experience. It aims to optimize job-specific parameters as well as resource utilization. The scheduling system is able to dynamically learn and cluster information in a large dimensional parameter space and at the same time to explore new regions in the parameter's space. This self-organizing scheduling system may offer a possible solution for providing an effective use of resources for the off-line data-processing jobs.

Finding and optimizing efficient job-scheduling policies in large distributed systems, which evolve dynamically, is a challenging task. It requires the analysis of a large number of parameters describing the jobs and the time-dependent state of the system. In one approach, the job-scheduling task in distributed architectures is based on self-organizing neural networks. The use of these networks enables the scheduling system to dynamically learn and cluster information in a high-dimensional parameter space. This approach may be applied to the problem of distributing off-line data processing. These jobs need random access to very large amounts of data, which are assumed to be organized and managed by distributed federations of OODB (object-oriented database) systems. Such a scheduling system may also help manage the way data are distributed among regional centers as a function of time, making it capable of providing useful information for the establishment and execution of data replication policies. A hybrid combination of neural networks and expert systems was tried by Apolloni, Zamponi, and Zanaboni (2000). Fdez-Riverola and Corchado (2003) used unsupervised learning for prediction of parameters during the learning process.

### SYSTEM DESCRIPTION

Case data that are presented to an expert system are usually stored in a case database. A data-transformation

module encodes such cases in a suitable way to be learned by neuronal networks. This module performs as follows:

First, it transforms or preprocesses the data so that the components with equal scopes or a hierarchy is formed. It has been shown that discretization of data (i.e., preprocessing of data in to several subdivisions) makes the neural network converge faster (Abu Bakar, Jaaman, Majid, Ismail & Shamsuddin, 2003). At the same time, hierarchy in the data can be maintained.

Second, the transformation module encodes the data into a binary input pattern because some neural networks, as, for example, the competitive learning model (Rumelhart & Zipser, 1985) processes only binary inputs. To do this, the intervals of the components are subdivided into different ranges. These ranges are adapted according to the distribution of the components. So, every component of a vector is represented by the range its value belongs to. Depending on the kind of representation, the ranges could be encoded locally.

With the transformed data, different neuronal networks with unsupervised learning algorithms, such as competitive learning (Rumelhart & Zipser, 1985), ART, and Kohonen, are trained. These networks have the ability to adapt their internal structures (i.e., weights) to the structure of the data. In a rule-generation module, the structures learned by the neuronal networks are detected, examined, and transformed into expert systems rules. These rules can be inspected by a human expert and added to an expert system. When a case is presented to the expert system, the system first tries to reason with the rules that have been acquired from the human expert to produce a suitable diagnosis. If this fails to produce a diagnosis, the new rules produced by the process described can be used. If the case can be handled in such a way, all steps of the reasoning process may be inspected by and explained to a user of the system.

If the system, however, is not able to produce a suitable diagnosis in this way, be it, that data is missing, or the input is erroneous or no rule fits the data, since such a case has not been considered while building the knowledge base, the expert system can turn the case over to the networks. The networks, with their ability to associate and generalize, search for a most suiting case that has been learned before. The diagnosis that has been associated with that case is then returned as a possible diagnosis.

### PROPOSED ARCHITECTURE

In recent years, neural networks have been extensively used to simulate human behavior in areas such as vision, speech, and pattern recognition. In large scales, they

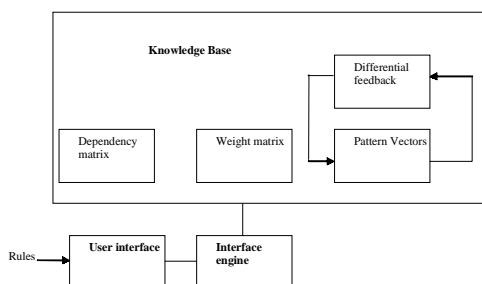


perform the recognition and classification tasks better than human beings. Neural nets can find a relation by making a match between known inputs and outputs in a pool of a large data. The performance of these networks can be improved by providing a differential feedback from the output to the input

Expert systems have been used in conjunction with neural network technology to eliminate the time-consuming part of constructing and debugging the knowledge base. Neural networks can be used to recognize patterns, such as financial or sensory data, which are then acted on by the rules of an expert system. An expert system trains the neural network that in turn generates rules for an expert system. The use of an artificial neural network greatly simplifies the knowledge-engineering task because it allows the system itself to construct the knowledge base from a set of training examples. Because the correct output of the expert system is known for the given set of inputs, a supervised learning algorithm is used to train the system.

Compared to rule-based expert systems, connectionist expert systems give a better model of reasoning. These models can be applied to any type of decision problems, especially when the creation of if-then rules is not possible, or the information is contradictory. A connectionist expert system usually contains three major parts: a knowledge base, an inference engine, and the user interface. The knowledge base is a problem-dependent part that contains expert knowledge. The connectionist expert system database is composed of neurons and connections among them. The inference engine is a problem-independent driver program which is responsible for reasoning. The user interface is a link between the inference engine and the external user. The architecture of a connectionist expert system is shown in Figure 1. The weight matrix stores the rules that are translated in to the input-output dependency in the neural network through the dependency matrix and the pattern vector. The differential feedback connects a part of the output to the input and reduces the training period as well as the errors in pattern matching.

Figure 1. An expert system with DANN-based learning



## DATA PROCESSING

The Kohonen network, consisting of two layers, is used here. The input layer has  $n$  units representing the  $n$  components of a data vector. The output layer is a two-dimensional array of units arranged on a grid. The number of the output units is determined experimentally. Each unit in the input layer is connected to every unit in the output layer, with a weight associated. The weights are initialized randomly, taking the smallest and the greatest value of each component of all vectors as boundaries. They are adjusted according to Kohonen's learning rule (Kohonen, 1984). The applied rule uses the Euclidean distance and a simulated Mexican-hat function to realize lateral inhibition. In the output layer, neighboring units form regions, which correspond to similar input vectors. These neighborhoods form disjoint regions, thus classifying the input vectors.

The automatic detection of this classification is difficult because the Kohonen algorithm converges to an equal distribution of the units in the output layer. Therefore, a special algorithm, the so-called U-matrix method, is used to detect classes that are in the data (Ultsch & Siemon, 1990).

In summary, by using this method, structure in the data can be detected as classes. These classes represent sets of data that have something in common.

## RULE EXTRACTION

As a first approach to generate rules from the classified data, a well-known, machine-learning algorithm, ID3, is used (Ultsch & Panda, 1991). Although able to generate rules, this algorithm has a serious problem: It uses a minimization criterion that seems to be unnatural for a human expert. Rules are generated that use only a minimal set of decisions to come to a conclusion

A large number of parameters, most of them time dependent, must be used for the job scheduling in large distributed systems. The problem is even more difficult when not all of these parameters are correctly identified, or when the knowledge about the state of the distributed system is incomplete or is known to have a certain delay in the past.

## SCHEDULING DECISION

A "classical" scheme to perform job scheduling is based on a set of rules using part of the parameters and a list of empirical constraints based on experience. It

may be implemented as a long set of hard coded comparisons to achieve a scheduling decision for each job. In general, it can be represented as a function, which may depend on large numbers of parameters describing the state of the systems and the jobs. After a job is executed based on this decision, a performance evaluation can be done to quantify the same.

The decision for future jobs should be based on identifying the clusters in the total parameter space, which are close to the hyperplane defined in this space by the subset of parameters describing the job and the state of the system (i.e., parameters known before the job is submitted). In this way, the decision can be made evaluating the typical performances of this list of close clusters and choose a decision set that meets the expected-available performance-resources and cost. However, the self-organizing network has, in the beginning, only a limited "knowledge" in the parameter space, and exploring efficiently other regions is quite a difficult task.

In this approach for scheduling, the difficult part is not learning from previous experience, but making decisions when the system does not know (or never tried) all possible options for a certain state of the system. Even more difficult is to bring the system into a certain load state, which may require a long sequence of successive decisions, such as in a strategic game problem. The result of each decision is seen only after the job is finished, which also adds to the complexity of quantifying the effect of each decision. For this reason, a relatively short history of the previous decisions made by the system is also used in the learning process. A relatively long sequence of the decision (a policy) can be described with a set of a few points of decision history, which partially overlap. This means that a trajectory can be built in the decision space by small segments that partially overlap.

## **FUTURE TRENDS**

The present-day expert systems deal with domains of narrow specialization. For such systems to perform competently over a broad range of tasks, they will have to be given a wealth of knowledge. The next generation expert systems require large knowledge bases. This calls for crisp learning algorithms based on connectionist networks with higher order differential feedback. Improvements in the learning rate and stability of these algorithms to organize large data structures provide a good topic for research.

## **CONCLUSION**

The implementation of the system demonstrates the usefulness of the combination of a rule-based expert system with neural networks (Ultsch & Panda, 1991). Unsupervised-learning neural networks are capable of extracting regularities from data. Due to the distributed subsymbolic representation, neural networks are typically not able to explain inferences. The proposed system avoids this disadvantage by extracting symbolic rules out of the network. The acquired rules can be used like the expert's rules. In particular, it is therefore possible to explain the inferences of the connectionist system.

Such a system is useful in two ways. First, the system is able to learn from examples with a known diagnosis. With this extracted knowledge it is possible to diagnose new unknown examples. Second, the system has the ability to handle a large data set for which a classification or diagnosis is unknown. For such a data set, classification rules are proposed to an expert. The integration of a connectionist module realizes "learning from examples." Furthermore, the system is able to handle noisy and incomplete data. First results show that the combination of a rule-based expert system with a connectionist module is not only feasible but also useful. The system considered is one of the possible ways to combine the advantages of the symbolic and subsymbolic paradigms. It is an example to equip a rule-based expert system with the ability to learn from experience using a neural network.

## **REFERENCES**

- Abu Bakar, A., Jaaman, S.H., Majid, N., Ismail, N. & Shamsuddin, M. (2003). Rough discretisation approach in probability analysis for neural classifier. *ITSIM, 1*, 138-147.
- Apolloni, B., Zamponi, G., & Zanaboni, A. M. (2000). An integrated symbolic/connectionist architecture for parsing italian sentences containing pp-attachment ambiguities. *Applied Artificial Intelligence, 14*(3), 271-308.
- Fdez-Riverola, F., & Corchado, J. M. (2003). Forecasting system for red tides: A hybrid autonomous AI model. *Applied Artificial Intelligence, 17*(10), 955-982.
- Kohonen, T. (1984). *Self-organization and associative memory*. Berlin, Germany: Springer-Verlag.
- Kuo, R. J., Wu, P. C., & Wang, C. P. (2000). Fuzzy neural networks for learning fuzzy if-then rules. *Applied Artificial Intelligence, 14*(6), 539-563.

Manjunath, R., & Gurumurthy, K. S. (2002). Information geometry of differentially fed artificial neural networks. *IEEE TENCON'02*, Beijing, China.

Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence*, 18, 203-226.

Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess end-games. In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (pp. 463-482).

Rumelhart, D. E., & Zipser, D. (1985). Feature discovery by competitive learning. *Cognitive Science*, 9, 75-112.

Ultsch, A., & Panda, P. G. (1991). Die Kopplung Konnektionistischer Modelle mit Wissensbasierten Systemen. *Tagungsband Experteny stemtage Dortmund* (pp. 74-94). VDI Verlag.

Ultsch, A., & Siemon, H. P. (1990). Kohonen's self organizing feature maps for exploratory data analysis. *Proceedings of International Neural Networks* (pp. 305-308). Kluwer Academic Press.

## KEY TERMS

**Artificial Intelligence (AI):** A research discipline whose aim is to make computers able to simulate human abilities, especially the ability to learn. AI is separated as neural net theory, expert systems, robotics, fuzzy control systems, game theory, and so forth.

**Connectionist Expert System:** Expert systems that use artificial neural networks to develop their knowledge bases and to make inferences are called connectionist expert systems. A classical expert system is defined with IF-THEN rules, explicitly. In a connectionist expert system, training examples are used by employing the generalization capability of a neural network, in which the network is coded in the rules of an expert system. The neural network models depend on the processing elements that are connected through weighted connections.

The knowledge in these systems is represented by these weights. The topology of the connections are explicit representations of the rules.

**Expert System:** An expert system is a computer program that simulates the judgment and behavior of a human or an organization that has expert knowledge and experience in a particular field. Typically, such a system contains a knowledge base containing accumulated experience and a set of rules for applying the knowledge base to each particular situation that is described to the program.

**Kohonen Feature Map:** It is basically a feed forward/feedback type neural net. Built of an input layer (i.e., the neuron of one layer is connected with each neuron of another layer), called "feature map." The feature map can be one or two dimensional, and each of its neurons is connected to all other neurons on the map. It is mainly used for classification.

**Neural Network:** A member of a class of software that is "trained" by presenting it with examples of input and the corresponding desired output. Training might be conducted using synthetic data, iterating on the examples until satisfactory depth estimates are obtained. Neural networks are general-purpose programs, which have applications outside potential fields, including almost any problem that can be regarded as pattern recognition in some form.

**Supervised Learning:** This is performed with feed forward nets where training patterns are composed of an input vector and an output vector that are associated with the input and output nodes, respectively. An input vector is presented at the inputs together with a set of desired responses, one for each node. A forward pass is done and the errors or discrepancies, between the desired and actual response for each node in the output layer, are found. These are then used to determine weight changes in the net according to the prevailing learning rule.

**Unsupervised Learning:** A specific type of a learning algorithm, especially for self-organizing neural nets such as the Kohonen feature map.

# High Quality Conceptual Schemes

Esko Marjoma

University of Joensuu, Finland

## FRAMEWORK FOR CONCEPTUAL MODELING

The questions of quality may be divided into four distinct classes, namely, ontological, epistemological, value-theoretical, and pragmatic. However, there are plenty of important problems to the solutions which have bearings on the different classes. Some of the problems are very tricky, and we shall explore two of them: (1) How does the basic ontology affect the form and content of the resulting conceptual model? and (2) What is the status of formalization in pragmatics? There are good reasons to believe that the answers to these two questions may also answer the other ones.

Conceptual modeling has been characterized in various ways, but the most central feature of it is twofold: (a) We model concepts by concepts, and (b) by “concepts” we mean “entities, the role of which is to carry the sameness of different tokens of definite entity types.” In practice, the following principles serve as general guidance for conceptual modeling (see Table 1).

It is useful to consider modeling processes as consisting of successive stages. The first thing we have to do in any definite modeling process is to *explicate* the tasks of modeling (i.e., we have to clearly express the goal of our modeling activity in question). In addition, we also have to explicate the *use* of the desired conceptual model and that of the conceptual schema. One tries to build the desired conceptual model (and the conceptual schema) in respect to the use of the model (and that of the conceptual schema).

Table 1. Principles of conceptual modeling

<p><i>P1 The conceptualization principle:</i> Only conceptual aspects of the Universe of Discourse (UoD) should be taken into account when constructing a conceptual schema.</p> <p><i>P2 The 100 Percent principle:</i> All the relevant aspects of the UoD should be described in the conceptual schema.</p> <p><i>P3 The correspondence condition for knowledge representation:</i> The <i>modellens</i> should be such that the recognizable constituents of it have a one-to-one correspondence to the relevant constituents of the <i>modellum</i>.</p> <p><i>P4 The invariance principle:</i> Conceptual schema should be constructed on the basis of such entities found in the UoD that are invariant during certain time periods within the application area.</p> <p><i>P5 The principle of contextuality:</i> Conceptual schema should be constructed on the basis of contextually relevant entities belonging to the UoD. This construction should be made according to some of the principles of abstraction levels using appropriate model constructs in order to uncover mutual relevance between different conceptual subschemata.</p>
--

After the explication phase, we need to describe the new information about the *modellum*. If the modellers need more information than just their perceptions concerning the *modellum*, they can get it, for instance, from the people living in the house, from libraries, and so forth. However, in order to get useful information, they have to describe the needed information as accurately as possible.

In addition to the description of the new information, we also need to get all the available information about the *modellum*. This is the proper *information acquisition* phase, after which we shall *analyze* the *received* information; it is essential here that we speak of *received* information, because we cannot be sure if we have all the information sent to us.

In order to construct a conceptual model of the application area, we often need to condense the information—not all the information—only the analyzed information. Then, the construction and the development of the conceptual model will be based only on that condensed information. After we have developed a conceptual model, we have to construct a physical representation of the conceptual model (in a form of a conceptual schema) using a language most appropriate for fulfilling the tasks in question. The technical realization of the conceptual schema and the information base will then be based on this physical representation.

However, there are other more basic prerequisites for the construction of the conceptual schemata (and hence for the technical realization of it), namely, the background factors that almost in *any* case affect the modeling process. They are presented in Table 2.

We may now ask, “What is relevant in information modeling processes to get high-quality technical realizations?” Obviously, each of the aforementioned stages (i) through (x). But it is also relevant to take into account different kinds of affecting background factors, such as those presented in Table 2.

Items (vii) to (xiii), especially, concern the question, “How does the basic ontology affect the form and content of the resulting conceptual model?” and items (i) to (xii) concern the question, “What is the status of formalization in pragmatics?” In the following sections, we shall consider these questions in a bit more detail.



Table 2. Background factors affecting modeling processes

- |        |   |
|--------|---|
| (i)    | modelers' profiles - for example, whether there are any multi-media experts among the modelers; |
| (ii)   | determinator – for example, the experts' profiles, paradigms, authorities;                      |
| (iii)  | practical constraints - for example, environment, time, money, tools;                           |
| (iv)   | skills of the users - for example, acquaintance with computers;                                 |
| (v)    | arbitrary conventions - for example, systems of marking;  |
| (vi)   | empirical facts - for example, particular constraints concerning some definite UoD;             |
| (vii)  | hypotheses - for example, generalization;   |
| (viii) | the ways of analyzing - for example, classification;  |
| (ix)   | the ways of abstracting - for example, aggregation, axiomatization;                             |
| (x)    | considerations of simplicity – for example, is the conceptual structure easily visualizable;    |
| (xi)   | considerations of fruitfulness - for example, is the method applicable elsewhere;               |
| (xii)  | idealizations - for example, estimation;  |
| (xiii) | metaphysical presuppositions - for example, ontological commitments.                            |

### CONSEQUENCES OF THE BASIC ONTOLOGY CHOICE

It seems that the more invariant an entity is, the more abstract it is. So, we should try to generalize this by examining whether there are some very basic (i.e., abstract) concepts that can be used to construct some general frames for considering different conceptual schemata.

According to Kangassalo (1983), the basic connections between different concepts and conceptual submodels are the relation of intentional containment and auxiliary (i.e., factual) relations. This strongly involves the account that there are certain concepts that are more basic than others. In other words, structuring conceptual schemata calls for some set of the so-called model constructs.

A semantically abstract model concept is characterized by Kangassalo (1983) as a concept the properties of which are defined only partially in the following way: (a) The extension of the concept is undefined or its definition specifies only the type of the elements of the reference class on a high level of abstraction (i.e., only some of the properties of the elements of the reference class are specified); and (b) the intension of the concept does not contain any factual concepts or, in addition to the set of nonreferential concepts, it contains some concepts that refer to an abstract model object. In other words, the intension of the concept does not contain any completely defined factual concepts. Some examples of such model concepts are type, entity, flow, process, event, and state.

According to Kangassalo (1983), model concepts can be regarded as primitive structuring elements the

use of which direct the modeling process by forcing the designer to complete the semantics of model concepts in order to get a completely defined conceptual model. In other words, if the designer wants to use the model concepts “entity,” “process,” and “event” as building blocks for a conceptual model, then he has to add factual semantics to all instances of these model concepts in the conceptual model.

A semantically abstract model, construct is characterized by Kangassalo (1983) as “a conceptual construct which contains only absolutely abstract concepts or semantically abstract model concepts” (p. 248). For brevity, it is usually called a model construct. Kangassalo gives the following examples of model constructs: system, hierarchy, network, schema, theory, and algebra.

“How to do we construct the set of model concepts?” or, to put it differently, “What is the basis on which we choose the model concepts out of the model constructs?” is a difficult problem. The set of model constructs, namely, may be extremely large, including an arbitrary collection of general nouns, such as *entity*, *process*, *event*, *state*, *flow*, *system*, *hierarchy*, *network*, *schema*, *theory*, *frame*, *object*, *substance*, *property*, *relation*, *act*, *disposition*, *ability*, *regularity*, *cause*, *explanation*, *function*, and so forth. This problem is closely interrelated to the problem of abstraction in conceptual modeling.

There are different conceptions of what ontology is. In philosophy, a general account of the issue can be stated thus: “Ontology aims to answer at least three questions, namely, (1) What there is? (2) What is it that there is? (3) How is that there is? These general ontological questions can be rephrased to apply to concepts: (1') What are concepts? (2') What stuff are they made of? (3') How can they exist?” (Palomaki, 1994, pp. 23-24).



In information modeling, there is a more specialized meaning of the word *ontology*, such as, especially, in Gruber (1993):

*“An ontology is a formal, explicit specification of a shared conceptualization. “Conceptualization” refers to an abstract model of phenomena in the world by having identified the relevant concepts of those phenomena. Explicit means that the type of concepts used, and the constraints on their use are explicitly defined. Formal refers to the fact that the ontology should be machine readable. Shared reflects that ontology should capture consensual knowledge accepted by the communities.”*

Although, according to Gruber (1993), ontology is a specification of a conceptualization, we may say, in general, that *conceptualizations are not possible without a correct basic ontology*. So, Gruberian ontologies are specifications of conceptualizations, which, in turn, should be built on correct, basic ontology. One good example of basic ontology can be found, for instance, in Sowa (2000) and at <http://www.jfsowa.com/ontology/toplevel.htm> where Sowa describes top-level categories and relations between them.

The World Wide Web Consortium has published a document (W3C, 2002) dealing with the requirements for a Web ontology language. The considerations are applicable also to conceptual modeling languages. In the document, the design goals describe general motivations for the language that do not necessarily result from any single use case. First, there is a description of eight design goals for an ontology language. For each goal, there is also a description of the tasks it supports and explains the rationale for the goal. One central goal is *shared ontologies*: “Ontologies should be publicly available and different data sources should be able to commit to the same ontology for shared meaning. Also, ontologies should be able to extend other ontologies in order to provide additional definitions” (Berners-Lee, Calilliay, & Groff, 2002).

The use cases and design goals in W3C (2002) motivate a number of requirements for a Web ontology language. The requirements, however, described in the document also seem to be essential to any ontologically correct language. Each requirement includes a short description and is motivated by one or more use cases or design goals.

However, before choosing any ontology, it is common to choose some suitable *ontology language(s)*. But before that, we have to evaluate different needs in knowledge representation, reasoning, and exchange of information (Corcho & Gomez-Perez, 2000).

## BE CAREFUL WITH FORMALISMS!

Most people cannot think formally. This fact yields some restrictions, or, at least, some requirements concerning desired conceptual schemata.

In developmental psychology, there is a conception that from age 14 to 16, people reach the stage of so-called formal operations. In this stage, intelligence is demonstrated through the logical use of symbols related to abstract concepts. Surprisingly, it has been shown that only one third of high school graduates in industrialized countries obtain formal operations.

We may talk about formal thinking which can be marked by the ability to systematically generate and work with larger spaces of possibilities, including possibilities that are quite abstract. Inhelder and Piaget tested for formal thinking by asking children and adolescents to design and conduct scientific experiments—for instance, experiments to determine what determines the period of a pendulum, or what factors affect the bending of rods that vary in shape, length, size, material, and so on. But thinking explicitly about your values and your course in life, and comparing them with other possible values and other possible courses in life, also qualifies as formal thinking (see Campbell, 2002.)

As Campbell (2002) noticed, Piaget did suggest that beyond formal operations, there are postformal operations, or “operations to the  $n^{\text{th}}$  power.” Inevitably these would be of a highly specialized nature, and might be found in the thinking of professional mathematicians or experts in some other fields. An early example of “operations to the  $n^{\text{th}}$  power” is Piaget’s statement that constructing axiomatic systems in geometry requires a level of thinking that is a stage beyond formal operations “one could say that axiomatic schemas are to formal schemes what the latter are to concrete operations” (Piaget, 1950, p. 226).

It would be an interesting research project to study whether the nature of expertise is due to the possession of schemas that guide perception and problem solving and how these schemas are dependent on the ability to use formal operations.

A crucial question is, What impact does the ability (or the lack of it) to use formal operations have on the creation or interpretation of conceptual structures? For instance, it has been proven quite clear that there are real differences between different students in constructing and interpreting conceptual schemata. Even among the students of computer science, the differences between the quality of constructed conceptual schemata are substantial: During a course on conceptual modeling, at the Department of Computer Science, University of Joensuu, in the spring of 2002, there was a

task to construct a conceptual schema of an intelligent refrigerator, and the resulting plans varied greatly.

One way to handle with the issue is to develop a *logic of evaluation*, or, perhaps rather, a *logic of preference*. First of all, it should be noted that preference is always related to a subject. It is always somebody's preference. Preference is also always related to a certain instant or situation.

The first purely logical attempts to handle the problem of preference or betterness were made by Hallden (1957) and von Wright (1963a). But the notion of preference is central also in economics, and especially in econometrics, where it is investigated usually together with the notions of utility and probability. In philosophy these concepts are in the realm of value theory. Generally we can say that such concepts as, for instance, right and duty are deontological, while such concepts as good, bad, and better are axiological. To anthropological concepts belong such concepts as need, desire, decision, and motive.

The borders between these different disciplines are not altogether clear, but the study of deontological and axiological concepts should be based on the study of anthropological concepts. One essential difference between deontological and axiological concepts seems to be that axiological concepts are often comparative (we can speak about their "degrees"), while deontological concepts are not.

It is important to note that the concept of preference involves not only the axiological concept of betterness but also the anthropological concept of selection. The notion of betterness (or goodness) has different modes which are not all intrinsically related to preference. This concerns especially technical goodness. For example, if *x* is better than *y*, there seems to be no direct connection to the notion of preference. However, technical goodness may sometimes have a close relation to instrumental goodness. For example, if *x* is better than *y*, we prefer using *x*.

Table 3. Two kinds of preference

<p>1. Sometimes one thing is preferred over another because the former is supposed to be "better" than the latter. For instance, a person may prefer beer to white wine because of his or her appetite. In this case, the mode of preference is called <i>extrinsic</i>.</p> <p>2. There are also people who prefer beer simply because they like it better than wine. This kind of mode of preference is called <i>intrinsic</i>. There are also different types of preference and different classificatory bases. Consider the common features of things (or states of affairs) between which is a preference relation:</p> <p>2.1. Some instrument (or the use of it) may be preferred over another.</p> <p>2.2. One way of doing something may be preferred over another.</p> <p>2.3. Some state of affairs may be preferred over another.</p>
--

All modes of preference are somehow related to goodness, but we can distinguish between two kinds of preference (see Table 3).

But it will be enough to develop a formal theory that handles only states of affairs, because all the other types of preference can be expressed in terms of states of affairs.

In information modeling, both modes of preference are employed. Some kind of logic of "extrinsic preference" is used especially when choosing methods and tools for representing information. But first we should introduce a logic of "intrinsic preference," because the motives or reasons of someone preferring one state of affairs over another are not always known. One such a logic is von Wright's *Logic of Preference* (1963a), and a good starting point to develop a real system of evaluation is his book, *Varieties of Goodness* (1963b).

## CONCLUSION

Keeping in mind that the aim of conceptual modeling is to construct a representation of the UoD (which is regarded as a system of signs or signals). To *condense* the information concerning the UoD is to express as briefly as possible all relevant information concerning the constituents of the UoD. Most often we use some of the methods shown in Table 4.

Table 4. Methods of information condensation

<p><b>Axiomatization</b> Process of representing the UoD by giving just a few sentences (axioms, or basic truths) from which we can deduce other truths of the system (theorems); this representation is called an <i>axiom system</i></p> <p><b>Classification</b> [of the UoD's constituents] Grouping the constituents into classes on the basis of the discovery of common properties</p> <p><b>Generalization</b> Process of arriving at a general notion from the individual instances belonging to a class</p> <p><b>Formalization</b> [of an axiom system] Step by which we add to the system such rules that are needed in proving the theorems of the system; <i>formalized axiom systems</i> can be regarded as the ideal cases of condensation</p> <p><b>Idealization</b> Process of constructing a representation that ignores some less relevant aspects of the UoD</p> <p><b>Abstraction</b> Process of separating a relevant partial aspect of a collection of entities</p>
---

## REFERENCES

- Berners-Lee, T. J., Calilliau, & Groff, J. F. (1992). The World Wide Web. *Computer Networks & ISDN Systems*, 25(4), 454-459.
- Campbell, R. L. (2002). Jean Piaget's genetic epistemology: Appreciation and critique. Retrieved April, 2002, from <http://hubcap.clemson.edu/campber/piaget.html>
- Corcho, O., & Gomez-Perez, A. (2000). Evaluating knowledge representation and reasoning capabilities of ontology specification languages. *Proceedings of the ECAI 2000 Workshop on Applications of Ontologies and Problem-Solving Methods*, Berlin, Germany. Retrieved June 2004, from <http://delicias.dia.fi.upm.es /WORKSHOP/ECAI00/schedule.html>
- Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5, 199-220.
- Hallden, S. (1957). On the logic of "better." *Library of Theoria*, No. 2. Uppsala.
- Kangassalo, H. (1983). Structuring principles of conceptual schemas and conceptual models. In J. A. Bubenko, Jr. (Ed.), *Information modeling* (pp. 223-307). Lund, Sweden: Studentlitteratur.
- Palomaki, J. (1994). From concepts to concept theory: Discoveries, connections, and results. *Acta Universitatis Tamperensis, ser. A, vol. 416*. Tampere, Finland: University of Tampere. Finland.
- Piaget, J. (1950). *Introduction l'pistmologie gntique. I: La pense mathmatique*. Paris: Presses Universitaires de France.
- Sowa, J. F. (2000). *Knowledge representation: Logical, philosophical, and computational foundations*. Pacific Grove, CA: Brooks Cole.
- W3C. (2002, March 7). Requirements for a Web ontology language. W3C working draft. Retrieved August, 2003, from <http://www.w3.org/TR/2002/WD-webont-req-20020307/>
- Wright, G. H. von. (1963a). *The logic of preference*. Edinburgh, Scotland: Edinburgh University Press.
- Wright, G. H. von. (1963b). *The varieties of goodness*. London: Routledge.

## KEY TERMS

**Conceptual Modeling:** Process of forming and collecting conceptual knowledge about the Universe of Discourse (UoD), and documenting the results in the form of a *conceptual schema*.

**Conceptual Schema:** A completely or partially time-independent description of a portion of the (real or postulated) world in the sense that a conceptual schema contains the definition of all concepts, and all relationships between concepts, allowed to be used in the description of that portion of the world.

**Entity:** Anything that can be conceived as real and can be named.

**Explication:** The act of making clear or removing obscurity from the meaning of a word, symbol, expression, and such.

**Modellens:** The constructed model, which may consist of different kinds of representation, such as sentences, figures, or letters.

**Modeller:** The modeling subject, the designer, a student, a group of agents, a camera, and so forth.

**Modellum:** The object to be modeled, the entity of interest, the universe of discourse, and so forth.

**Ontology:** In philosophy, the theory of being. In information systems design, ontology defines the terms used to describe and represent some area of knowledge.

**Pragmatics:** A subfield of linguistics that studies how people comprehend and produce communicative acts in concrete situations.

**Universe of Discourse (UoD):** A collection of all those entities that have been, are, or ever might be in a selected portion of the real world or the postulated world.

# The Information Quality of Databases

**InduShobha Chengalur-Smith**  
*University at Albany, USA*

**M. Pamela Neely**  
*Rochester Institute of Technology, USA*

**Thomas Tribunella**  
*Rochester Institute of Technology, USA*

## INTRODUCTION

A database is only as good as the data in it. Transaction-processing systems and decision-support systems provide data for strategic planning and operations. Thus, it is important to not only recognize the quality of information in databases, but also to deal with it. Research in information quality has addressed the issues of defining, measuring and improving quality in databases; commercial tools have been created to automate the process of cleaning and correcting data; and new laws have been created that deal with the fallout of poor information quality. These issues will be discussed in the next sections.

## BACKGROUND

Data or information quality began to achieve prominence during the total quality management movement in the 1980s. However, it has relevance to individual decision making as well as to organizational data. With the advent of the World Wide Web, the concept of information overload became universal, and the notion of information quality had instant appeal. As is typical in an emerging field, there are several research themes in the information quality literature. In this section, we highlight topics in some of the major research areas.

Although data quality is now widely perceived as critical, there is less agreement on exactly what is meant by high-quality information. Clearly it is a multidimensional construct whose measure is very much context dependent. A substantial amount of work has been devoted to identifying the dimensions of data quality and their interrelationships. Early work on the dimensions of data quality looked at pairs of attributes, such as accuracy and timeliness, and the trade-offs between them (e.g., Ballou & Pazer, 1995).

A more comprehensive study that attempted to capture all the dimensions of information quality was conducted by Wang and Strong (1996). They conducted a

two-stage survey and identified about 180 attributes of data quality, which they combined into 15 distinct dimensions. These dimensions were further organized into four categories: intrinsic quality, contextual quality, representation, and accessibility.

There have been several follow-on studies that focus on subsets of these dimensions. For instance, Lee and Strong (2003) examined five of these dimensions and concluded that when data collectors know why the data is being collected, it leads to better quality data. Another avenue for research has been in developing formulae and metrics for each of these dimensions (e.g., Pipino, Lee, & Wang, 2002).

Yet another branch of research in information quality considers information to be a product of an information manufacturing process (Ballou, Wang, Pazer, & Tayi, 1998). Thus, the output of an information system can be tracked much like a manufacturing product, allowing for the application of quality-control procedures to improve the quality of information products. This concept has been elaborated on by Shankaranarayan, Ziad, and Wang (2003) by creating an information product map that represents the flow and sequence of data elements that form the information product.

## MAIN FOCUS

Wang, Lee, Pipino, and Strong (1998) listed the steps an organization must take to successfully manage information as a product, which has led to the development of several software tools and methods for modeling and measuring data quality.

Not long ago, the majority of the software tools that fell into the data quality area depended on comparing data in databases to something else (Neely, 1998). Broadly defined, data quality tools provide three functions. They audit the data at the source; clean and transform the data in the staging area; and monitor the extraction, transformation, and loading process. Some of the



tools are an extension of data validation rules that should be a part of the database design but are frequently ignored in legacy systems. These tools, commonly referred to as *auditing tools*, involve comparing the data to a known set of parameters, which might be a minimum value, a set of dates, or a list of values. Once the data is run through the tool, results are routinely examined manually, a very time- and labor-consuming process.

Another category of tools, data cleansing tools, originally began as name and address tools. The core functionality of these tools is the ability to parse, standardize, correct, verify, match, and transform data. Ultimately, these tools are designed to produce accurate lists that can be used or sold.

Acquisitions and mergers have consolidated what was once a fragmented set of software tools that addressed specific roles in the data-quality process into integrated tools that can be used throughout the process of arriving at high-quality data. Companies such as Trillium Software, which once provided tools focusing on the audit function, have now expanded their functionality into the cleansing arena. Ascentials Software acquired Vality, primarily a data cleansing tool, and Ardent, primarily an auditing tool, providing it with the ability to monitor the entire process. Some of the earliest techniques in automating the process involved data mining techniques, discovering patterns in the data and reverse-engineering the process by suggesting business rules. WizRule has taken an early lead in using this technology.

Data profiling is being heralded as the new generation of data quality tools. Tools that provide this functionality, such as Data Quality Suite, claim that the inherent problem with auditing data is that it prevents the user from seeing the big picture. The focus is on the data elements, whereas with data profiling, the focus is on what the data should look like, based on aggregates of the data.

The metadata associated with data expands as the data is used for secondary purposes, such as for data warehousing, which in turn supports analytical tools, such as data mining or On-Line Analytical Processing (OLAP) tools. What was once a data dictionary, describing the physical characteristics of the data, such as field type, becomes a complex web of physical definitions, quality attributes, business rules, and organizational data. Some of the newest tools are those that attempt to manage metadata. Commercial metadata tools fall into two categories: tools for working with metadata, and centralized repositories. The metadata tools attempt to interpret technical metadata so that the business users can understand it (Levin, 1998). Central repositories are based on industry standards, in an effort to allow third-party warehouse and analysis tools to interface with

them. Many systems attempt to combine the technical specifications with business information (Platinum Technology, 2000); however, there is still room for improvement.

Information quality has also been studied as a contributing factor to information systems (IS) success. A widely accepted model of IS success holds that the quality of information obtained from a system impacts satisfaction with, and ultimately the success of, the system (DeLone & McLean, 2003). In an empirical study of the factors affecting the success of data warehouse projects, data quality is shown to have a significant impact (Wixom & Watson, 2001). However organizational, project and technical success did not result in high-quality data in the warehouse. A study by Strong, Lee, and Wang (1997) identified the major reasons for poor-quality data and suggested ways to improve the data collection and information dissemination processes.

The major impact of poor-quality data is on decision making. To make users aware of the quality of the data they are dealing with, the use of data tags has been proposed (Wang & Madnick, 1990). Databases could be tagged at varying levels of granularity and using different dimensions of data quality. For instance, by recording the time a data item was last updated, a user could assess the currency of the data. Creating separate tags for each data item would be very resource intensive. Alternatively, a tag for each record or field could be used to represent the quality of the information stored there. When considering what information the tag should contain, it is important to remember that the data will be used in the context of a specific situation. Thus, the quality of the data may be sufficient in one instance, but not in another. The "fitness-for-use" concept continues to be an issue when applying data tags. Regardless, the data tags form a metadata about the quality of the data.

Clearly, keeping the metadata on information quality up to date could itself be a Herculean task. Before embarking on such an ambitious project, it would be worthwhile to examine the impact of providing this information about the quality of the data. If decision makers were unable to use this information, for any reason, whether it was because it was not in the appropriate format or because it created information overload, then the effort of producing this metadata would be futile.

Some exploratory research (Chengalur-Smith, Ballou, & Pazer, 1999) has shown that the format in which the data quality information is provided does play a role in the use of the information. Fisher, Chengalur-Smith, & Ballou, (2003) determined that experienced decision makers were more likely to use the data quality information if they did not have expertise in the data domain and if they did not feel time pressure. Further research will refine these findings.



## The Information Quality of Databases

An important aspect of information quality is trust. If users do not trust the source of the data, they are unlikely to use the data or any resulting information product. There has been a lot of research devoted to interpersonal trust and, in a business setting, trust among partnering organizations. When designing information systems, the credibility of the data has to be established. Tseng and Fogg (1999) discuss the two kinds of errors that could occur when consuming information: the “gullibility” error, in which users believe information they should not, and the “incredulity” error, in which users fail to trust data that is actually credible. Tseng and Fogg emphasize that the responsibility for reducing the incredulity error rests with the designers of database and information systems.

Wathen and Burkell (2002) indicate that users first assess the surface and message credibility of a system before they evaluate its content. Surface credibility can be assessed through the interface design characteristics, and message credibility includes information quality characteristics such as source, currency, and relevance. The threshold credibility required would vary from user to user and would depend on the context, motivation, expertise, and so forth, of the user. A major challenge for system designers is creating systems that have universal access. The Holy Grail for such systems is that is they can be built to adapt to individual users’ preferences (Stephanidis, 2001). Thus, the data in these systems needs to be custom tailored so that the presentation pace and format is matched to users’ knowledge-acquisition styles.

A major concern for databases is security. Often, databases contain confidential information (e.g., financial data). The trade-offs between security and accessibility has been studied by several researchers. One approach has been to establish different levels of access and provide only aggregate statistics to the general public but allow authorized individuals to access the entire dataset for decision making purposes (Adam & Wortmann, 1989). A related approach proposed by Muralidhar, Parsa, and Sarathy (1999) suggests perturbing the data with random noise in such a way that the relationships between the attributes do not change. Their process ensures that querying the database with the modified data will give accurate results but that the original confidential data will not be disclosed.

Information auditing is another aspect of information quality that is gaining in popularity. An information product is the output from an information system that is of value to some user. In 2002, the U.S. Congress enacted the Sarbanes-Oxley Act “to protect investors by improving the accuracy and reliability of corporate disclosures made pursuant to the securities laws.” The act was a direct reaction to the financial malfeasance that

occurred with corporations such as Enron and Global Crossing. It requires management to report “all significant deficiencies in the design or operation of internal controls which could adversely affect the issuer’s ability to record, process, summarize, and report financial data.” This raises the level of importance regarding data quality concerns and will require accountants as well as other professionals to receive more training on data-quality issues. In addition, the act requires auditors to assess internal controls and attest to its adequacy and effectiveness at maintaining accurate and reliable financial data. Analyzing internal control procedures requires auditors to think about systems and focus attention on the business processes that produce or degrade data quality.

According to the American Institute of Certified Accountants, data storage, data flows, and information processes, as well as the related internal control systems, must be regularly documented and tested. Data quality can be measured in terms of a risk assessment, which estimates the probability that data integrity has been maintained by the control environment. A violation of internal control, such as an improper authorization or a breakdown in the segregation of duties, should require an immediate response from management to correct the problem and maintain the organization’s confidence in the quality of the data.

As part of a data audit, companies may be required to assess the quality of the reports or the other information products they generate. In the context of a relational database, an information product is generally obtained by combining tables residing in the database. Although it may be feasible to assess the quality of the individual tables in a database, it is impossible to predict all the ways in which this data may be combined in the future. Furthermore, a given combination of tables may have a very different quality profile from the constituent tables, making it impossible to surmise the quality of information products with any degree of confidence. Ballou et al (2004) have proposed a method for estimating the quality of any information product based on samples drawn from the base tables. They developed a reference table procedure that can be used as a gauge of the quality of the information product, regardless of the combination of relational algebraic operations used to create the product.

## FUTURE TRENDS

One of the technological trends that will affect data quality is markup languages. The creation of markup languages such as eXtensible Markup Language (XML) will allow for the rapid communication of data between

organizations, systems, and networks. XML allows users to define an application-specific markup language and publish an open standard in the form of a taxonomy of tags; XML-based tags that can be used to identify grains of data for financial, scientific, legal and literary applications.

For example, eXtensible Business Reporting Language (XBRL) is an XML-based open standard being developed by XBRL International Inc. The XBRL taxonomy is a royalty-free standard that is supported by a not-for-profit consortium of approximately 200 companies and agencies. Accordingly, the XBRL standard has been adopted by all major producers of accounting information systems (AIS) and enterprise resource planning (ERP) systems. These systems will produce financial information in an XBRL-compliant format so that the information can be published, retrieved, exchanged, and analyzed over the Internet. This common platform, which supports the business reporting process, will improve the ease and increase the quantity of financial data being received by managers and investors.

Currently, XBRL is being used by the Securities Exchange Commission (SEC) to accept financial reports that contain data in XBRL-compliant form. New ways of testing data quality and reliability using markup technology will have to be developed by investors, accountants, regulators, business executives, and financial analysts. Furthermore, XBRL will reduce latency by increasing the speed with which financial information is reported to the market. In the future, XBRL-compliant systems could theoretically change the reporting cycle from periodic to real time. However, real-time on-line financial reporting will require the development of real-time data-assurance methods.

## CONCLUSION

This article covered some key research topics in information quality. We began with the attempts to capture the many dimensions of data quality and their inter-relationships. We also focused on the technology that has been developed to automate data cleansing. The impact of data quality on the success of and satisfaction with information systems was also discussed. Future developments in databases may include data tags that are metadata about the quality of the data. On an individual level, the effectiveness of this metadata is still being investigated. Some preliminary research has shown that the data tags are most helpful to users who are experienced but not necessarily familiar with the subject domain of the database. Finally, the Sarbanes Oxley Act will have wide reaching consequences for all data quality.

## REFERENCES

- Adam, N. R., & Wortmann, J. C. (1989). Security control methods for statistical databases: A comparative study. *ACM Computing Surveys*, 21(4), 515-556.
- Ballou, D. P., Chengalur-Smith, I. N., & Wang, R. Y. (2004). *Quality estimation for information products in relational database environments*.
- Ballou, D. P., & Pazer, H. L. (1995). Designing information systems to optimize the accuracy timeliness trade-offs. *Information Systems Research*, 6(1), 51-72.
- Ballou, D. B., Wang, R. Y., Pazer, H. L., & Tayi, G. K. (1998, April). Modeling information manufacturing systems to determine information product quality. *Management Science*, 44(4), 462-484.
- Chengalur-Smith, I. N., Ballou, D. P., & Pazer, H. L. (1999). The impact of data quality information on decision making: An exploratory analysis. *IEEE Transactions on Knowledge and Data Engineering*, 11(6), 853-864.
- DeLone & McLean. (2003). The DeLone and McLean model of information systems success: A ten year update. *Journal of Management Information Systems*, 19(4), 9-30.
- Fisher, C., Chengalur-Smith, I. N., & Ballou, D. P. (2003). The impact of experience and time on the use of data quality information in decision making. *Information Systems Research*, 14(2), 170-188.
- Lee, Y., & Strong, D. (2003). Knowing why about data processes and data quality. *Journal of Management Information Systems*, 20(3), 13-39.
- Levin, R. (1998). Meta matters. *Information Week*, 18-19.
- Muralidhar, K, Parsa, R., & Sarathy, R. (1999). A general additive data perturbation method for database security. *Management Science*, 45(10), 1399-1415.
- Neely, P. (1998). Data quality tools for data warehousing: A small sample survey. *Proceedings of the International Conference on Information Quality*, Cambridge, MA.
- Pipino, L., Lee, Y. W., & Wang, R. Y. (2002). Data quality assessment. *Communications of the ACM*, 45(4), 211-218.
- Platinum Technology. (2000). *Putting metadata to work in the warehouse*. Computer Associates White Paper.
- Shankaranarayan, G., Ziad, M., & Wang, R. Y. (2003). Managing data quality in dynamic decision environment: An information product approach. *Journal of Database Management*, 14(4).

## The Information Quality of Databases

Stephanidis, C. (2001). Adaptive techniques for universal access. *User Modeling and User-Adapted Interaction*, 11, 159-179.

Strong, D. M., Lee, Y. W., & Wang, R. W. (1997). 10 potholes in the road to information quality. *Computer*, 30(8), 38-46.

Tseng, H., & Fogg, B. J. (1999). Credibility and computing technology. *Communications of the ACM*, 42(5), 39-44.

Wang, R. Y., Lee, Y. W., Pipino, L., & Strong, D. M. (1998). Manage your information as a product. *Sloan Management Review*, 39(4), 95-105.

Wang, R. Y. & Strong, D. (1996). Beyond accuracy: What data quality means to data consumers. *Journal of Management Information Systems*, 12(4), 5-34.

Wathen, C. N., & Burkell, J. (2002). Believe it or not: Factors influencing credibility on the Web. *Journal of the American Society for Information Science and Technology*, 53(2), 134-144.

Wixom, B. H., & Watson, H. J. (2001). An empirical investigation of the factors affecting data warehousing success. *MIS Quarterly*, 25(1), 17-41.

## KEY TERMS

**Auditing:** a systematic process of objectively obtaining and evaluating evidence regarding assertions about data and events to ascertain the degree of compliance with established criteria.

**Data Tags:** Quality indicators attached to fields, records, or tables in a database to make decision-makers aware of the level of data quality.

**Fitness for Use:** Describes the many variables that need to be considered when evaluating the quality of an information product.

**Information Product:** The output of an information manufacturing system; it implies stages of development, such as information suppliers, manufacturers, consumers, and managers.

**Information Quality:** The degree to which information consistently meets the requirements and expectations of the knowledge workers in performing their jobs.

**Internal Control:** A system of people, technology, processes, and procedures designed to provide reasonable assurance that an organization achieves its business process goals.

**Metadata:** Data about data; that is, data concerning data characteristics and relationships.

**Risk Assessment:** A set of procedures whose purpose is to identify, analyze, and manage the possibility that data quality has been degraded due to events and circumstances such as a breakdown in internal controls, system restructuring, changes in organizational culture, or alterations in the external environment.

# Integration of Data Semantics in Heterogeneous Database Federations

H. Balsters

University of Groningen, The Netherlands

## INTRODUCTION

Modern information systems are often distributed in nature; data and services are spread over different component systems wishing to cooperate in an integrated setting. Information integration is a very complex problem and is relevant in several fields, such as data reengineering, data warehousing, Web information systems, e-commerce, scientific databases, and B2B applications. Information systems involving integration of cooperating component systems are called *federated information systems*; if the component systems are all databases then we speak of a *federated database system* (Rahm & Bernstein, 2001; Sheth & Larson, 1990). In this article, we will address the situation where the component systems are so-called legacy systems; i.e., systems that are given beforehand and which are to interoperate in an integrated single framework in which the legacy systems are to maintain as much as possible their respective autonomy.

A huge challenge is to build federated databases that respect so-called global transaction safety; i.e., global transactions should preserve constraints on the global level of the federation. In a federated database (or FDB, for short) one has different component databases wishing to cooperate in an integrated setting. The component systems are often legacy systems: They have been developed in isolation before development of the actual federated system (they remain to be treated as autonomous entities). Legacy systems were typically designed to support local requirements; i.e., with local data and constraints, and not taking into account any future cooperation with other systems. Different legacy systems may also harbour different underlying data models, subject to different query transaction processing methods (flat files, network, hierarchical, relational, object-oriented, etc.). Getting a collection of autonomous legacy systems to cooperate in a single federated system is known as *the interoperability problem*.

The general term *mediation* (Wiederhold, 1995) was founded to address the problem of interoperability. A federated database (FDB) can be seen as a special kind of mediation, where the mediator acts as a DBMS-like interface to the FDB application.

A mediator is a global service to link local data sources and local application programs. It provides integrated information while letting the component systems of the federation remain intact. Typical mediator tasks include:

- accessing and retrieving relevant data from multiple heterogeneous sources,
- transforming retrieved data so that they can be integrated,
- integrating the homogenized data.

The mediator provides a database-like interface to applications. This interface gives the application the impression of a *homogeneous, monolithic database*. In reality, however, queries and transactions issued against this interface are translated to queries and transactions against underlying component database systems. Mediation is typically realized by defining a suitable uniform data model on the global level of the federation; such a global uniform data model is targeted at resolving the (ontological) differences in the underlying component data models (Rahm & Bernstein, 2001). Best candidates, on the conceptual level, are *semantic data models*, e.g., UML/OCL (Warmer & Kleppe, 2003), with the aim to define a data model powerful enough to harbour:

- rich data structures,
- an expressive language for operations,
- a rich constraint language.

This article will discuss some problems related to semantic heterogeneity, as well as offer an overview of some possible directions in which they can be solved.

## BACKGROUND

Data integration systems are characterized by an architecture based on a global schema and a set of local schemas. There are generally three situations in which the data integration problem occurs. The first is known as global-as-view (GAV), in which the global schema is defined directly in terms of the source schemas. GAV



systems typically arise in the context where the source schemas are given, and the global schema is to be derived from the local schemas. The second situation is known local-as-view (LAV), in which the relation between the global schema and the sources is established by defining every source as a view over the global schema. LAV systems typically arise in the context where the global schema is given beforehand, and the local schemas are to be derived in terms of the global schema. The third situation is known as *data exchange*, characterized by the situation that the local source schemas, as well as the global schema, are given beforehand; the data integration problem then exists in establishing a suitable mapping between the given global schema and the given set of local schemas (Miller, Haas, & Hernandez, 2000). An overview of data integration concentrating on LAV and GAV can be found in Lenzerini (2002); articles by Abiteboul and Douschka (1998), Grahne and Mendelzon (1999), and Halevy (2001) concentrate on LAV, whereas Cali, Calvanese, De Giacom, and Lenzerini (2002), Türker and Saake (2000), and Vermeer and Apers (1996) concentrate on GAV. Our article focuses on legacy problems in database federations in the context of GAV; in particular, we deal with the situation that a preexisting collection of autonomous component databases is targeted to interoperate on the basis of mediation. The mediator is defined as a virtual database on the global level and is aimed at faithfully (i.e., on the basis of completeness and consistency) integrating the information in the original collection of component databases.

A major problem that we will address in this article is that of so-called *semantic* heterogeneity (Bouzeghoub & Lenzerini, 2001; Hull, 1997; Rahm & Bernstein, 2001; Vermeer & Apers, 1996). Semantic heterogeneity refers to disagreement on (and differences in) meaning, interpretation, or intended use of related data. The process of creation of uniform representations of data is known as *data extraction*, whereas *data reconciliation* is concerned with resolving data inconsistencies. Examples of articles concentrating on GAV as a means to tackle semantic heterogeneity in database federations are found in Balsters (2003), Cali et al. (2002), Türker and Saake (2000), and Vermeer and Apers. These articles concern the following topics: Cali et al. treats data integration under global integrity constraints; Türker and Saake concerns integration of local integrity constraints; and Vermeer and Apers abstracts from the relational model, as we do in this article, by offering a solution based on an object-oriented data model (Balsters, de By, & Zicari, 1993; Balsters & Spelt, 1998). This article differs from the aforementioned articles in the following aspects. In contrast to Cali et al., we also take local integrity constraints into account;

furthermore, our approach adopts an approach restricted to so-called sound views instead of exact ones. The article by Türker and Saake abstracts from problems concerning data extraction by assuming the existence of a uniform data model (pertaining to all participating local databases) in which all problems regarding semantic heterogeneity have been straightened out beforehand. Our article, in contrast, offers a treatment of data extraction and reconciliation in a combined setting and as an integral part of the mapping from local to global.

### OUR FOCUS: SEMANTIC HETEROGENEITY

The problems we are facing when trying to integrate the data found in legacy component frames are well known and extensively documented (Lenzerini, 2002; Sheth & Larson, 1990). We will focus on one of the large categories of integration problems coined as *semantic heterogeneity* (Balsters & de Brock, 2003a, 2003b; Bouzeghoub & Lenzerini, 2001; Hull, 1997; Vermeer & Apers, 1996). Semantic heterogeneity refers to disagreement on (and differences in) meaning, interpretation, or intended use of related data. Examples of problems in semantic heterogeneity are *data extraction* and *data reconciliation*. The process of creation of uniform representations of data is known as data extraction, whereas data reconciliation is concerned with resolving data inconsistencies. The process of data extraction can give rise to various inconsistencies due to matters pertaining to the *ontologies* (Rahm & Bernstein, 2001) of the different component databases. Ontology deals with the connection between syntax and semantics, and how to classify and resolve difficulties and classification between syntactical representations on the one hand and semantics providing interpretations on the other hand.

Integration of the source database schemas into one encompassing schema can be a tricky business due to: *homonyms* and *synonyms*, *data conversion*, *default values*, *missing attributes*, and *subclassing*. These five conflict categories describe the most important problems we face in dealing with integration of data semantics. We will now shortly describe in informal terms how these problems can be tackled.

- Conflicts due to *homonyms* are resolved by mapping two same name occurrences (but with different semantics) to different names in the integrated model. *Synonyms* are treated analogously, by mapping two different names (with the same semantics) to one common name. In the sequel, we will use the abbreviations **hom** (**syn**) to indicate that we



have applied this method to solve a particular case of homonym (synonym) conflicts.

- Conflicts due to *conversion* arise when two attributes have the same meaning, but their domain values are differently represented. For example, an attributes salary (occurring, say, in two classes Person and Employee in two different databases DB1 and DB2, respectively) could indicate the salary of an employee, but in the first case the salary is represented in the currency dollars (\$), while in the latter case the currency is given in euros (•). One of the things we can do is to convert the two currencies to a common value (e.g., \$, invoking a function `convertTo$`). Another kind of conversion can occur when the combination of two attributes in one class have the same meaning as one attribute in another class. We will use the abbreviation **conv** to indicate that we have applied this method to solve a particular case of a representation conflict.
- Conflicts due to *default values* occur when integrating two classes and an attribute in one class is not mentioned in the other (similar) class, but it could be added there by offering some suitable default value for all objects inside the second class. As an example, consider an attribute part-time (occurring, say, in class Person in database DB1) could also be added to some other related class (say, Employee in database DB2) by stipulating that the default value for all objects in the latter class will be 10 (indicating full-time employment). We will use the abbreviation **def** to indicate that we have applied this method to solve a particular case of a default conflict.
- Conflicts due to *differentiation* occur when the integration of two classes calls for the introduction of some additional attribute in order to discriminate between objects originally coming from these two classes, e.g., in the case of conflicting constraints. Consider as an example the classes Person (in DB1) and Employee (in DB2). Class Person could have as a constraint that salaries are less than 1,500 (in \$), while class Employee could have as a constraint that salaries are at least 1,000 (in •). These two constraints seemingly conflict with each other, obstructing integration of the Person and the Employee class to a common class, say, Integrated-Person. However, by adding a discriminating attribute `dep` to Integrated-Person indicating whether the object comes from the DB1 or from the DB2 database, one can differentiate between two kinds of employees and state the constraint on the integrated level in a suitable way. We will use the abbreviation **diff** to indicate that we have

applied this method to solve a particular case of a differentiation conflict.

Problems pertaining to semantic heterogeneity in database integration (dealing with applications of **hom**, **syn**, **conv**, **def**, and **diff**) can all be treated in a uniform and systematic manner. We refer the interested reader to Sheth and Larson (1990), Hull (1997), Lenzerini (2002), and Balsters (2003a, 2003b) for exact details on this subject, but we can give an indication of how such a database integration can take place.

The basic idea is to arrive at an encompassing, global database, say, DBINT (from *integrated database*), in combination with a particular set of constraints. Data extraction will be performed by providing a global schema in which the local data coming from local databases can be uniformly represented, while data reconciliation has been performed by resolving constraint conflicts by suitable use of differentiation on the global level.

The strategy to integrate a collection of legacy databases into an integrated database DBINT is based on the following principle:

*An integrated database DBINT is intended to hold (and maintain to hold!) exactly the “union” of the data in the source databases in the original collection of local databases.*

In particular, this means that a given arbitrary update offered to the global database should correspond to a unique specific update on a (collection of) base table(s) in the original collection of local databases (and vice versa). We have to keep in mind that any update, say,  $t$ , on the global level is an update on a virtual (nonmaterialized) view; hence, it has to correspond to some concrete update, say,  $\mu$ , implementing  $t$  on the local level of the component frame. This is the *existence* requirement. There is also a *uniqueness* requirement that has to be met. Suppose, for example, that some update  $t$  on the global level corresponds to two different updates, say,  $\mu$  and  $\mu'$ , on the local level. This would lead to an undesirable situation related to the fact that in a database federation, existing legacy components *remain* to exist after integration and also are assumed to respect existing applications (i.e., those application running on the legacy databases irrespective of the fact that such a legacy database happens to also participate in some global encompassing federation). Such an existing application could involve a query on a legacy database, which could give two different query results, depending on which of the two concrete updates  $\mu$  and  $\mu'$  is performed on the local level. A database view, such as DBINT, satisfying both the

existence and uniqueness criteria as described above, is called an *exact view*. The general view update problem is treated extensively in Abiteboul, Hull, and Vianu (1995). In our setting in which we wish to construct DBINT as an exact view over the component databases, the results offered in Abiteboul et al. (1995) entail that the universe of discourse of the original collection of component databases and the universe of discourse of the integrated database DBINT are, in a mathematical sense, *isomorphic*. Only in this way will we not lose any information when mapping the legacy components to the integrated database (and vice versa). The interested reader is referred to Balsters (2003a, 2003b) for more details on this subject.

### FUTURE TRENDS

Data reconciliation and data extraction are now fairly well understood on the theoretical level. A topic acquiring recent attention is integration of *database constraints* and introduction of so-called *federation constraints* on the global level of the federation (Balsters, 2003b; Türker & Saake, 2000). Integration of local transactions within database federations (closely related to the problem of constraint integration) is still in demand of more research.

Federated database systems are now also appearing on a more commercial level. As a prominent, and promising, product, we mention IBM DB2 Information Integrator (cf. <http://www.ibm.com>, date of download May 2003), enabling users to build integrated databases without migrating data from local to global sources.

Many of the techniques referred to in this article are also applied in settings related to data federations, of which we mention data warehousing and enterprise application integration. Both of the latter-mentioned application areas are heavily embedded in data integration problems.

### CONCLUSION

A data federation provides for tight coupling of a collection of heterogeneous legacy databases into a global integrated system. Two major problems in constructing database federations concern achieving and maintaining consistency and a uniform representation of the data on the global level of the federation. The process of creation of uniform representations of data is known as data extraction, whereas data reconciliation is concerned with resolving data inconsistencies. A particular example of a data reconciliation problem in database federa-

tions is the integration of integrity constraints occurring within component legacy databases into a single global schema.

We have described an approach to constructing a global integrated system from a collection of (semantically) heterogeneous component databases based on the concept of exact view. A global database constructed by exact views integrates component schemas without loss of constraint information; i.e., integrity constraints available at the component level remain intact after integration on the global level of the federated database.

### REFERENCES

- Abiteboul, S., Hull, R., & Vianu, V. (1995). *Foundations of databases*. Reading, MA: Addison-Wesley.
- Abiteboul, S., & Douschka, O. (1998). Complexity of answering queries using materialized views. In *Proceedings of the Seventeenth ACM SIGACT-SIGMID-SIGART Symposium on Principles of Database Systems*. New York: ACM Press.
- Balsters, H. (2003). Modeling database views with derived classes in the UML/OCL-framework. In *Lecture Notes in Computer Science: Vol. 2863. «UML» 2003 Sixth International Conference*. Berlin, Germany: Springer.
- Balsters, H., de By, R. A., & Zicari, R. (1993). Sets and constraints in an object-oriented data model. In *Lecture Notes in Computer Science: Vol. 707. Proceedings of the Seventh ECOOP*. Berlin, Germany: Springer.
- Balsters, H., & de Brock, E. O. (2003a). An object-oriented framework for managing cooperating legacy databases. In *Lecture notes in computer science: Vol. 2817. Ninth International Conference On Object-Oriented Information Systems*. Berlin, Germany: Springer.
- Balsters, H., & de Brock, E. O. (2003b). Integration of integrity constraints in database federations. *Sixth IFIP TC-11 WG 11.5 Conference on Integrity and Internal Control in Information Systems*. Norwell, MA: Kluwer.
- Balsters, H., & Spelt, D. (1998). Automatic verification of transactions on an object-oriented database. In *Lecture notes in computer science: Vol. 1369. Sixth International Workshop on Database Programming Languages*. Berlin, Germany: Springer.
- Bouzeghoub, M., & Lenzerini, M. (2001). Introduction. *Information Systems*, 26.
- Cali, A., Calvanese, D., De Giacomo, G., & Lenzerini, M. (2002). Data integration under integrity constraints. In

*Lecture Notes in Computer Science: Vol. 2348. Proceedings of CAISE 2002.* Berlin, Germany: Springer.

Grahne, G., & Mendelzon, A. O. (1999). Tableau techniques for querying information sources through global schemas. In *Lecture Notes in Computer Science: Vol. 1540. Proceedings of ICDT'99.* Berlin, Germany: Springer.

Halevy, A. Y. (2001). Answering queries using views: A survey. *VLDB Journal*, 10.

Hull, R. (1997). Managing semantic heterogeneity in databases. In *Proceedings of the Sixteenth ACM SIGACT-SIGART Symposium on Principles of Database Systems.* New York: ACM Press.

Lenzerini, M. (2002). Data integration: A theoretical perspective. In *ACM PODS'02.* ACM Press.

Miller, R. J., Haas, L. M., & Hernandez, M. A. (2000). Schema mapping as query discovery. In *Proceedings of The 26th VLDB Conference.* San Mateo, CA: Morgan Kaufmann.

Rahm, E., & Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *VLDB Journal*, 10.

Sheth, A. P., & Larson, J. A. (1990). Federated database systems for managing distributed, heterogeneous and autonomous databases. *ACM Computing Surveys*, 22.

Türker, C., & Saake, G. (2000). Global extensional assertions and local integrity constraints in federated schemata. *Information Systems*, 25(8).

Vermeer, M., & Apers, P. G. M. (1996). The role of integrity constraints in database interoperation. In *Proceedings of the 22nd VLDB Conference.* San Mateo, CA: Morgan Kaufmann.

Warmer, J. B., & Kleppe, A. G. (2003). *The object constraint language* (2nd ed.). Boston: Addison-Wesley.

Wiederhold, G. (1995). Value-added mediation in large-scale information systems. In *Proceedings of IFIP Data Semantics.* Norwell, MA: Kluwer.

## KEY TERMS

**Data Exchange:** The situation that the local source schemas, as well as the global schema, are given beforehand; the data integration problem then exists in establishing a suitable mapping between the given global schema and the given set of local schemas.

**Data Extraction:** The process of creation of uniform representations of data in a database federation.

**Data Reconciliation:** The process of resolving data inconsistencies in database federations (such as constraint conflicts).

**Database Federation:** A database federation provides for tight coupling of a collection of heterogeneous legacy databases into a global integrated system. Main problem is achieving and maintaining consistency and a uniform representation of the data on the global level of the federation.

**GAV:** Global-As-View, in which the global schema is defined directly in terms of the source schemas. GAV systems typically arise in the context where the source schemas are given, and the global schema is to be derived from the local schemas.

**Interoperability Problem:** Getting a collection of autonomous legacy systems to cooperate in a single federated system.

**LAV:** Local-As-View, in which the relation between the global schema and the sources is established by defining every source as a view over the global schema. LAV systems typically arise in the context where the global schema is given beforehand and the local schemas are to be derived in terms of the global schema.

**Mediation:** A global service to link local data sources and local application programs, thus providing the integrated information on the global level, while letting the component systems of the federation to remain intact.

**Ontology:** Ontology deals with the connection between syntax and semantics, and how to classify and resolve difficulties and classification between syntactical representations on the one hand and semantics providing interpretations on the other hand.

# Integrative Document and Content Management Systems' Architecture

**Len Asprey**

*Practical Information Management Solutions Pty Ltd., Australia*

**Rolf Green**

*OneView Pty Ltd., Australia*

**Michael Middleton**

*Queensland University of Technology, Australia*

## INTRODUCTION

### Purpose

This paper discusses the benefits of managing digital content (business documents and Web content) within the context of an integrative information systems architecture. This architecture incorporates database management, document and Web content management, integrated scanning/imaging, workflow, and data warehousing technologies.

### Business Context

The ubiquitous use of digital content (such as office documents, e-mail, and Web content) for business decision-making makes it imperative that adequate systems are in place to implement management controls over digital content repositories. The traditional approach to managing digital content has been for enterprises to store it in folder structures on file or Web servers. The content files stored within folders are relatively unmanaged, as there are often inadequate classification and indexing structures (taxonomies and metadata), no adequate version control capabilities, and no mechanisms for managing the complex relationships between digital content. These types of relationships include embedded or linked content, content renditions, or control over authored digital documents and published Web content.

In some cases enterprises have achieved a form of management control over hard-copy documents that are records of business transactions by using database applications to register, track, and manage the disposal of physical files and documents. These types of file or document "registers" do not provide adequate controls over the capture, retrieval, and accessibility to digital content.

This deficiency has led to many organizations seeking solutions, such as document management systems, to manage digital business content. Document management systems have generally been implemented to meet regulatory compliance within the context of document record-keeping requirements or management of digital archive collections. Otherwise, they have been implemented as solutions for managing specific types of content objects, such as ISO9001 quality management system documentation, engineering drawings, safety documents, and similar.

More recently, organizations have sought to acquire Web content management systems with the view to providing controls over digital content that is published to Web sites. The imperative for such a solution may be a commercial one, motivated by product-to-market visibility, customer service, and profitability. There may also be a response to compliance needs, motivated by managing Web content in the context of "record keeping" to satisfy regulatory or governance requirements.

The methodology of implementing document or Web content management systems has often been based on a silo approach, with more emphasis on tactical business imperatives than support for strategic enterprise information architecture initiatives. For example, organizations may attempt a Web content management solution without taking into full account digital documents that may be used to create content outside the constraints of Web-compatible formats such as XML-defined, but which are subsequently required for publication. Thus, document and Web content management may be viewed as discrete solutions, and business applications may be implemented without an integrative approach using workflow and systems for managing both business documentation and Web content.

Another example of a silo approach is the deployment of database solutions without cognizance of document or Web content management requirements. For example, organizations may deploy a solution for man-



aging contracts, including database application capabilities for establishing the contract, recording payments and variations, and managing contract closure. However, the management of contract documents may not be viewed as an integral part of the application design, or the workflow review and approval, or managing the published contract materials on Web sites. The result is that users often miss vital information rather than manually relate data retrieved through a number of separate applications.

There are compelling reasons for organizations, as they address the constructs of enterprise information architecture, to consider the management of digital content within the context of an integrative approach to managing business documents and Web content. The strategic rationale for such an approach encompasses the following types of business imperatives:

- Customer satisfaction is a key commercial driver for both business and government: in the case of the commercial sector, the need to attract and retain customers, and in the public sector, the need to support government initiatives directed at taxpayer benefits. Organizations are adopting strategic approaches such as single view of customer and one-source solution for customer information, invoking the use of information knowledge management tools.
- Speed and quality of product to market is another major business driver. The rapid adaptation of the WWW and e-commerce systems to support online business transactions opens markets to global competition. Commercial enterprises are not only required to deliver product to market rapidly, but also within quality management constraints, to attract and retain customers.
- Regulatory imperatives, such as Sarbanes-Oxley in the United States (U.S. Congress, 2002) have introduced new measures for creating greater transparency within organizations, which impact corporate governance and require disclosure with real-time reporting requirements.

The enterprise information architecture would include information policy, standards and governance for the management of information within an organization, and provide supporting tools in the form of an integrative information systems architecture as the platform for managing information. An integrative systems architecture would provide a platform that enables businesses to meet the challenges of both commercial and regulatory imperatives, benefit from reusable information, and provide a coherent view of relevant information enterprise-wide to authorized users.

In respect to document and Web content management, an integrative document and content management (IDCM) model (Asprey & Middleton, 2003) offers a framework for unification of these components into an enterprise information architecture. The model features the management of both documents and Web content within an integrative business and technology framework that manages designated documents and their content throughout the document/content continuum and supports record-keeping requirements.

## **SCOPE**

The core IDCM elements that address document and Web content management requirements (capturing content, reviewing/authorizing content, publishing content, and archival/disposal) comprise:

- Integrated document and Web publishing/content management capabilities.
- Integration of document-imaging capabilities.
- Recognition technologies, such as bar codes, to assist with capturing document information or conversion of image data to text as a by-product of scanning/imaging.
- Enterprise data management capabilities.
- Workflow.

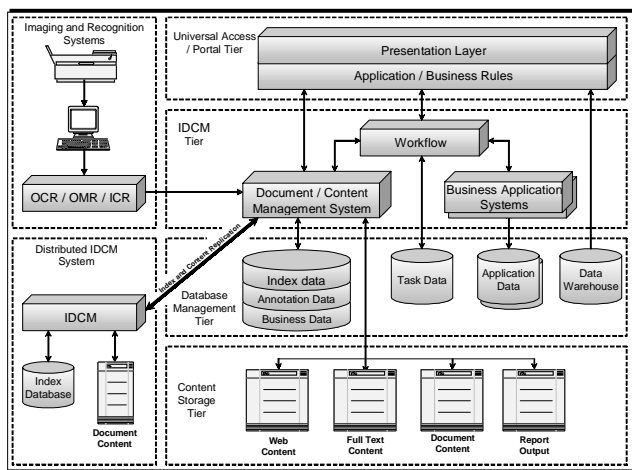
However, when determining requirements within the context of process improvement initiatives that help to address business imperatives (such as customer satisfaction, product to market, and regulatory compliance), these capabilities might be supported by other technologies. This technology support may help businesses to achieve an integrative systems architecture for deployment of innovative and integrated solutions.

- Universal access/portal, which allows users to invoke functions and view information (including digital content) via a Web-based interface.
- Integration with business systems, such as enterprise resource planning (ERP) systems, human resource systems, financial systems, and vertical line of business systems.

These types of capabilities, when combined, augment an integrative systems architecture to support the development of solutions that take advantage of digital content in managed repositories. Users that access business information then have the confidence that they are accessing, retrieving, and printing the most current



*Figure 1. Information systems architecture: document/content management*



digital content. This confidence can aid decision making, as end users may not then be required to access physical copies of documents, which can be both cumbersome and time-consuming due to customer expectations on speed of service in a modern technology environment.

The following section discusses those features of an integrative information systems architecture that would support a requirement for business users to gain rapid access to the most up-to-date digital content via an interface that is simple and intuitive.

## SYSTEM FEATURES

### Schematic

Figure 1 provides a schematic of database management within the context of IDCM and supporting technologies, such as universal interface (e.g., portal), and interfaces to business applications.

### Universal Access/Portal

Universal access/portal applications are used within enterprises to allow the viewing and interpretation of information from a number of sources, providing a single view of information. These sources of information may include data stored within application databases, data stored in warehouses, as well as digital documents and other content stored within document and content management systems. The evolution of universal access/portal applications is to encapsulate not only multisys-

tem data but to provide a single application interface for all enterprise systems.

Corporate portals may be differentiated by application (Collins, 2003) or level (Terra & Gordon, 2003), but they are typically aimed to enhance basic Intranet capability. This may be achieved by provision of facilities such as personalization, metadata construction within enterprise taxonomy, collaborative tools, survey tools, and integration of applications. The integration may extend to external applications for business-business or business-customer transactions or for derivation of business intelligence by associated external and internal information.

Universal access/portal applications include the capability of understanding how the information in each system is related to the business process and the rules by which the data can be combined. They provide different views of information depending on the user's needs and allow information to be utilized to greater effect. Just as importantly they provide the controls by which the information capture of the enterprise can be managed to allow the support of both industry and compliance requirements.

These applications employ business intelligence to provide the integrative framework to support utilization of information across the enterprise. This framework allows enterprises to provide on demand reporting of information and to use the information actively in business applications rather than the more passive approach of printed reporting.

## DOCUMENT/CONTENT MANAGEMENT

Document management applications are used within enterprises to implement management controls for digital and physical document objects. These objects may include office documents, e-mail and attachments, images and multimedia files, engineering drawings, and technical specifications. The systems manage complex relationships between documents and provide capabilities to manage integrity, security, authority, and audit. Business classification and metadata capabilities support access, presentation, and disposal (Bielawski & Boyle, 1997; Wilkinson et al., 1998), thus supporting organizational knowledge management initiatives.

Web content management applications are implemented by organizations to manage digital content that is published to the Web, including Internet, intranet, and extranet sites. The functionality of Web content management applications can be summarized as content creation, presentation, and management (Arnold,

2003; Boiko, 2002; Robertson, 2003). Organizations are discerning the requirement to have integrated architectures for managing documents and Web content and to consider the implications of record-keeping requirements when seeking unified document and content solutions.

## **Document Imaging**

Document imaging systems are used to scan and convert hard-copy documents to either analog (film) or digital format (Muller, 1993). Film-based images can be registered and tracked within a document management system, similar to the way in which a paper document can be registered and tracked. Digital images can be captured into either a document or Web content management system via integration with the scanning system.

Digital imaging may involve black and white, grayscale, or color imaging techniques, depending on the business need, as the requirement may involve scanning a range of document types, e.g., physical copies of incoming correspondence, forms processing, or capture of color brochures.

Some systems allow users to search and view film images on their local computer by the film being scanned on demand. The speed of viewing film images is slow due to the need for the specified film roll to be loaded in the scanning device, for the film to be wound to the correct position, and then for the image to be scanned. Digital images may be stored on magnetic storage, allowing rapid recall by users and so supporting not only the records archive process but the demands of workflow and document management.

## **Recognition Systems**

Recognition systems, such as bar code recognition, optical mark recognition (OMR), optical character recognition (OCR), or intelligent character recognition (ICR), are often used to facilitate data capture when documents are scanned to digital image.

Systems can take advantage of encoding within bar codes, which can be intelligently encoded to assist with improving business processes. For example, a bar code may contain specific characters that identify a geographical region, product type, or details of a specific subject/topic of interest.

OMR systems are used to capture recognition marks on physical documents, such as bank checks, during digital scanning.

OCR and ICR technologies can allow text to be extracted from a document during digital scanning and conversion processes. The text might be an alphabetical string, or unique number, or words within a zoned area of a form. Depending on the quality of the hard-copy original document, much of the printed information might be convertible to text within acceptable error constraints.

OCR and ICR may also be used to convert existing digital documents, such as Tagged Image File (TIF) Format or Portable Document Format (PDF) documents, to a full text rendition capable of being searched for specific words or phrases and so increasing the ability to recall the primary file being the PDF format.

## **Enterprise Data Management**

Each of the components within an IDCM application is highly dependent on database technology for the storage of data, including the metadata relating to the various information objects, and often storage of the objects themselves. Software for managing the repositories must accommodate versioning, routing, security, and review and approval processes. It should also link to tailored interfaces and be able to deal with different renditions of documents, maintain associations within complex documents, and provide for complex queries. Information retrieval development in this area focuses on facilitating indexing and integrating taxonomies (Becker, 2003).

Many organizations use data warehouse applications to store archived records from various business systems. In such cases data may be accessible only through a separate user interface, therefore requiring manual steps to reference data to records within the originating and other systems. The implementation of data repositories within an integrated architecture allows cross-referencing and searching of corporate knowledge and information across the enterprise.

The improvement of data usage increases the number of data calls to the supporting databases and the need for data to be accessible across a wider geographic area. Data supporting document/content management, workflow, and other business systems may be required to be managed across a distributed architecture of replicated and cached data stores.

Integration of data warehouse and document/content management systems allows for various types of data, such as business reports, system logs, etc., to be stored as data or text files rather than within the data warehouse database. This integration allows storage planning based

on information life cycles, supporting better data management.

### **Workflow**

Workflow systems allow more direct support for managing enterprise business processes. These systems are able to automate and implement management controls over tasks from initiation of a process to its closure. Though many application systems can be said to have some level of workflow, they lack the flexibility that enables visual creation and alteration of processes by an enterprise workflow system. More importantly the enterprise workflow allows processes to flow across a number of systems rather than be isolated within a single application.

The drivers for implementing workflow may be to improve both efficiency (by automating and controlling work through a business process) and effectiveness (by monitoring the events work may pass through). Ideally, the design of such systems takes into account a unified analysis of enterprise information content (Rockley, Kostur, & Manning, 2003).

The implementation of workflow is often driven from a technology perspective rather than by business processes. Questions of flexibility and interoperability become the drivers for the project rather than enhancements to the business processes. The most marked effect of this is the amplification of bottlenecks within the business process when it is run through a workflow system.

Workflow systems provide the integrative framework to support business processes across the enterprise. This is done by providing a capability for the business easily to represent their businesses processes within the workflow system and by allowing the workflow to call the business functions specific to each system with the required data.

### **System Integration**

Given the business impetus towards improving customer service, such as optimization of product to market, in what are mostly highly volatile environments, there is a need for management to obtain rapid access to strategic, tactical, and operational information. Management information is typically stored in database applications, such as ERP systems, human resource systems, financial systems, and vertical line of business systems. However, information stored in databases often needs to be supplemented by copies of supporting documentation.

Integration between document/content repositories and operational/ administrative systems enables end users to access all information that is relevant to a particular matter. For example, copies of a contract may be required to support a contract variation, or a supporting invoice may be required to support payment processing.

### **REASONS FOR UTILIZATION**

The implementation of systems for managing digital and Web content in silo systems may only solve a tactical business requirement. Essentially, controls over content are implemented. However, silo approaches to the implementation of systems, while they perhaps solve tactical requirements, might not be the most strategic and innovative approach to help solve the key imperatives facing enterprises. Solutions that embrace integrated document and Web content management, combined with universal interface/portal applications and integration with business systems, may support better opportunities for business process improvement.

With respect to knowledge management, document and Web content management solutions support enterprise knowledge management initiatives, where the knowledge has been made explicit. The strategic approach to managing both data in databases and digital content using an integrative systems architecture may provide a more cohesive and coherent management of information, with the potential to add more value to knowledge management initiatives (Laugero & Globe, 2002). The use of the universal interface or portal will help to address usability issues with document and content management applications. While these systems contain a wide range of functionality, the end user may not need to invoke much of the functionality, and an enterprise portal-style interface approach allows better presentation and management of end-user functions and viewing capabilities.

Organizations may find that it is difficult to cost justify digital document management or Web content management applications based purely on information management notions of "better managing enterprise document collections." The investment in the technology may outweigh the perceived benefits. However, the approach of solving strategic management challenges using an integrative systems architecture to deliver end-to-end business applications may make it easier to support business justification. At the same time, the enterprise is able to secure a managed repository for documents and Web content to support information policy and planning.

Table 1. A summary of critical issues

Business Issues	Technology Issues
<b>Cost</b> The cost of an information systems architecture is likely to be higher than a tactical or silo solution.	<b>Infrastructure</b> The infrastructure within the organization may not be adequate for the deployment of the information systems architecture.
<b>Planning</b> The extent of planning required for the acquisition and implementation of a strategic enterprise platform is likely to be longer than that required for a tactical solution.	<b>Systems Integration</b> The integration between the components of the solutions may be jeopardized if technical specifications are not correctly and thoroughly defined or if the package selection process is not well-managed.
<b>Specifications</b> The extent of specifications required for a strategic enterprise platform is likely to be more extensive than that required for a tactical solution.	<b>Evolving Technology</b> The evolving nature of technology, including technology convergence, may impact the long-term rollout of the information systems architecture.
<b>Benefits Realization</b> Benefits of a strategic solution may not be realized as early as those for a tactical solution. However, the benefits may be more enduring.	<b>Security</b> The nature of the architecture, including integrated components, is likely to involve the development of detailed specifications to ensure seamless access to authorized information.
<b>Lack of Business Unit Buy-In</b> Autonomous business units may not see the benefit of a strategic enterprise solution, focusing instead on specific tactical and operational requirements.	<b>Disaster Recovery</b> Storage of large amounts of data made available on demand introduces the need for complex backup procedures and longer recovery times.

## CRITICAL ISSUES

Table 1 summarizes some of the critical issues that enterprises may need to consider when reviewing requirements for an information systems architecture that features integrative document and content management capabilities.

## CONCLUSION

The acquisition of enterprise document and Web content management systems might not be justified based only on notions of good information management or contribution to knowledge management. The likelihood of a document and content management project targeted at the enterprise being successful may be enhanced significantly by incorporating the capabilities of these systems into an integrative information systems architecture. This platform may then allow the business to initiate projects that deliver a wide range of business process improvement initiatives, all relying on a supporting and consistent enterprise platform and which provides easy access to authorized users of information. Thus, the platform becomes strategically positioned to support business imperatives in the strategic, tactical, and operational hierarchy of an organization and to allow the deployment of innovative and stable end-to-end business solutions.

## REFERENCES

- Addey, D., Ellis, J., Suh, P., & Thiemecke, D. (2002). *Content management systems*. Birmingham, UK: Glasshaus.
- Arnold, S.E. (2003). Content management's new realities. *Online*, 27(1), 36-40.
- Asprey, L., & Middleton, M. (2003). *Integrative document and content management: Strategies for exploiting enterprise knowledge*. Hershey, PA: Idea Group Publishing.
- Becker, S.A. (2003). *Effective databases for text & document management*. Hershey, PA: IRM Press.
- Bielawski, L., & Boyle, J. (1997). *Electronic document management systems: A user centered approach for creating, distributing and managing online publications*. Upper Saddle River, NJ: Prentice Hall.
- Boiko, B. (2002). *Content management bible*. New York: Hungry Minds.
- Collins, H. (2003). *Enterprise knowledge portals*. New York: American Management Association.
- Laugero, G., & Globe, A. (2002). *Enterprise content services: A practical approach to connecting content management to business strategy*. Boston: Addison-Wesley.

Muller, N.J. (1993). *Computerized document imaging systems: Technology and applications*. Boston: Artech House.

Robertson, J. (2003). *So, what is a content management system?* Retrieved March 18, 2004, from [http://www.steptwo.com.au/papers/kmc\\_what/index.html](http://www.steptwo.com.au/papers/kmc_what/index.html)

Rockley, A., Kostur, P., & Manning, S. (2003). *Managing enterprise content: A unified content strategy*. Indianapolis, IN: New Riders.

Terra, J., & Gordon, C. (2003). *Realizing the promise of corporate portals*. Boston: Butterworth-Heinemann.

United States Congress. (2002, July 30). *Public Law 107-204: Sarbanes-Oxley Act of 2002*. Retrieved July 16, 2004, from [http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?db name=107\\_cong\\_public\\_laws&docid=f:publ204.107.pdf](http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?db name=107_cong_public_laws&docid=f:publ204.107.pdf)

Wilkinson, R., Arnold-Moore, T., Fuller, M., Sacks-Davis, R., Thom, J., & Zobel, J. (1998). *Document computing: Technologies for managing electronic document collections*. Boston: Kluwer.

## KEY TERMS

**Content Management:** Implementation of a managed repository for digital assets such as documents, fragments of documents, images, and multimedia that are published to intranet and Internet WWW sites.

**Document Capture:** Registration of an object into a document, image, or content repository.

**Document Imaging:** Scanning and conversion of hard-copy documents to either analog (film) or digital image format.

**Document Management:** Implements management controls over digital documents via integration with standard desktop authoring tools (word processing, spreadsheets, and other tools) and document library functionality. Registers and tracks physical documents.

**IDCM:** Integrative document and content management.

**Portal:** User interface to a number of process and information sources.

**Recognition Technologies:** Technologies such as bar code recognition, optical character recognition (OCR), intelligent character recognition (ICR), and optical mark recognition (OMR) that facilitate document registration and retrieval.

**Workflow Software:** Tools that deal with the automation of business processes in a managed environment.



# Intension Mining

**Héctor Oscar Nigro**

*INCA/INTIA, Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina*

**Sandra Elizabeth González Císaro**

*Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina*

## INTRODUCTION

Knowledge discovery is defined as “the non trivial extraction of implicit, unknown, and potentially useful knowledge of the data” (Fayyad, Piatetsky-Shapiro, Smyth, & Uthurusamy, 1996, p. 6). According to these principles, the knowledge discovery process (KDP) takes the results just as they come from the data (i.e., the process of extracting tendencies or models of the data), and it carefully and accurately transforms them into useful and understandable information. To consider the discovery of knowledge useful, this knowledge has to be interesting (i.e., it should have a potential value for the user; Han & Kamber, 2001).

Current data mining solutions are based on decoupled architectures. Data mining tools assume the data to be already selected, cleaned, and transformed. Large quantities of data are required to provide enough information to derive additional knowledge (Goel, 1999). Because large quantities of data are required, an efficient process becomes essential.

With the idea of efficiency, intension mining was born. Gupta, Bhatnagar, and Wasan proposed the architecture and framework.

Intension mining arises as a framework that focuses on the user of the current KDP. The basic idea behind the concept of intension mining is to separate the user from the intricacies of the KDP and give him or her a single database management system (DBMS)-like interface to interactively mine for the required kind of knowledge. The user can plan the data mining needs beforehand and input them in the form of knowledge discovery schema (KDS). The system mines knowledge and presents the results in the required format. Intension mining leads to efficiency and makes the whole process more realistic, user-friendly, and, hence, popular (Goel, 1999). As a result, intension mining is a logical extension of incremental mining, with an oriented paradigm to the user, who establishes and conceives the requirements of the mining before the mining begins (Gupta, Bhatnagar, & Wasan, 2000a, 2000d).

## BACKGROUND

There are countless contributions to improve and understand KDP. The concept of a second-generation data mining system (Imielinski & Mannila, 1996; Virmani, 1998) involves rule generation, data-rule management, and rule postprocessing. Another extension includes providing users with the ability to remember past mining sessions. Virmani (1998) developed a design called discovery board, which provides a framework for DBMS-like environment supporting query language and APIs to build data mining applications.

Imielinski and Mannila (1996) proposed an evolution of KDP with an SQL-like interface for ad hoc mining. Meo, Psaila, and Ceri (1998) suggested architecture strongly coupled with an SQL server. Ganti, Gherke, and Pamakrishman (2000) developed DEMON, a system based on an incremental mining paradigm; with DEMON, it is possible to mine the entire data repository or some selected subset.

The work being done in the field of structured data mining and upcoming database ideas, such as Hippocratic databases, share various levels of commonality with the I-MIN model in their core ideas (Gupta et al., 2000c). An extensive and explicative coverage with other researches can be found in Gupta et al. (2000a) and Gupta, Bhatnagar, and Wasan (2001).

## INTENSION MINING: MAIN FEATURES

### General Characteristics

The basic proposal of intension mining is the separation of the mining’s requirements to prepare the data a priori and, as a result, to carry out the task of the mining in a more efficient and structured way.

Once planned, the objectives of data mining are kept in the form of a KDS. Besides capturing the requirements, the functionality of this scheme is to provide the user with a friendly interface, improving the user's productivity due to its understanding (Gupta et al., 2000a).

As the database framework's outlines contain the specifications of the relations, the KDS contains the specification of the mining requirements. The outline of knowledge discovery guides the selection, cleaning, transformation, and aggregation processes of the data before mining and, due to the readiness of the requirements in the KDS, the system is able to execute off-line pre-mining operations periodically. This information should be preserved in secondary storage in an appropriate form to be used to satisfy the mining queries performed by the user (Gupta et al., 2000a). Thus, the mining in the base can be carried out on the basis of demand, using this information.

An important characteristic of intension mining is that it perceives the KDP as a continuous process. Because the temporary aspect is captured by the operations of periodic premining, experimentation as well as monitoring is possible (Gupta et al., 2000c).

Intension mining, as DBMS architecture (for more detail on DBMS, see Elmasri & Narvathe, 2000), is built up in three phases:

- **Phase 1-Planning:** The aim of phase 1 is to evaluate the objectives of data mining and to formalize them as a KDS. As was previously mentioned, the user anticipates the mining requirements and specifies them in the aforementioned scheme during the planning phase. A clear understanding of the domain and the requirements of the mining help in designing a KDS, which directs the KDP. The type of knowledge to be mined in the database, the database to be mined, the transformation, the selection, the cleaning, the specific mining algorithm to be executed, and finally the presentation tools to be used are specified by the user in the KDS (Goel, 1999). The metadata is stored to be used in the accumulation and mining phases. Just as a good database outline design can efficiently satisfy the users' queries in most occasions, a well-thought-out KDS would be able to efficiently discover different types of knowledge in most instances (Gupta, 2000). At this level, security, backup, and recovery issues also arise in the database (Gupta et al., 2000a).
- **Phase 2-Accumulation:** Phase 2 starts after compilation of the KDS and continues until the user decides to drop the mining requirements (schema) altogether. During this accumulation phase, the incremental database is premined and aggregated in consultation with the metadata to yield knowledge

concentrate (KC), which stores the intermediate form of intended knowledge (Gupta et al., 2001).

The crucial parts of this level are the interaction between the database and the KDP to get to the registers of data and the maintenance of the KC in the secondary storage. The extraction of the KCs, starting from the incremental database, represents the intensive task of I/O in intension mining and it can endure several scannings of the data. Significantly, all these tasks are carried out off-line (Gupta et al., 2000a).

In conclusion, the presence of the trade-off that the KCs imply can be observed. Although KCs allow new functionality to be added, because the mining, when working on them is speeded up, they have a cost when occupying extra space. In the ideal case, all the tuples of the database will be the same, and so will the small structures; but, in the worst case, all the tuples are different; they can occupy a great quantity of space. One should then evaluate what is convenient for the user in each case.

- **Phase 3-Mining:** Mining is the final phase of the system. In general, during this step the KDP is intensive. The phase of mining begins when a user invokes a mining query in the user's interface (Goel, 1999). The user specifies the parameters when carrying out the query. This offers the user the freedom to explore the database and to experiment with the KDP. The query is processed, and the mining is done on the specific structures of data, which are kept in the KC (Gupta, 2000).

The mining process consults the KDS, and it executes the algorithm of the specified mining on the cumulative KCs during the accumulation phase. Finally, the results are presented by means of the presentation tool (Gupta et al., 2000a).

Those response times are also better because the I/O is avoided and is giving the possibility of carrying out an *exploratory analysis*, choosing the subset of data, or varying the parameters. As the mining is carried out on the KCs, it does not interfere with the operations of the database.

## I-MIN MODEL: AN INSTANTIATION OF INTENSION MINING

The pattern has been designed to support intension mining. Thus, it is developed in three layers, according to the basic ideas mentioned in Gupta et al. (2000a).

The architectural proposal emulates a DBMS-like environment for the managers, administrators, and end users in the organization. Knowledge management functions, such as sharing and reusing of the discovered

knowledge among the users and periodically updating the discovered knowledge are supported. The I-MIN model facilitates complete documentation and control of all the KDD endeavors in an organization. It helps in structuring and streamlining the KDD operations in an organization (Gupta, Bhatnagar, & Wasan, 2005).

## COMPARISON WITH THE TRADITIONAL PATTERN KDP

The model of process I-MIN provides the complete functionality of the traditional pattern KDP and a more structured focus of the process of knowledge discovery (Gupta, 2000).

- **Compatibility:** The model of process I-MIN captures completely the functionality of the traditional pattern KDP. The steps in the traditional pattern can be directly mapped to the steps of the process of the proposed pattern. Thus, the I-MIN pattern includes the traditional KDP pattern (Gupta et al., 2000a).
- **Added functionality:** The I-MIN pattern for the KDP supports experimentation and monitoring, which are desirable in explorations of both scientific and business data. The objectives of user-defined mining, stored in schema form, direct KDP. Periodic and systematic aggregations of data help achieve the functionality for experimentation and monitoring (Gupta et al., 2000a).  
For monitoring purposes, triggers can be set up and manipulated on a defined outline using the interface provided. The mining queries can also be formulated and executed by the steps of the process. Changing the perspective of the mining, the user can repeat the steps resulting in a discovery of exploratory and interactive knowledge for experimentation (Gupta et al., 2000a).
- **Formal focus:** In the traditional KDP pattern, several loosely couple independent processes are involved. The process pattern I-MIN integrates the premining operations and even link them to the analysis and aggregation of the data, to a certain degree. The strong merge during the accumulation phase imparts formalism and continuity to the whole process of knowledge discovery (Gupta et al., 2000a).  
The incorporation of the knowledge discovery administrator (KDA) and the planning of the operations of knowledge discovery give a formal “status” to the KDP in an organization (Gupta et al., 2000b).

## MODEL OF PROCESS I-MIN: FUNCTIONAL COMPONENTS

The implementation of the process pattern I-MIN essentially requires the development of the components of accumulation, mining, experimentation, and monitoring. Each component carries out a step in the process pattern. However, a combination of more than one component can be required to complete different functionalities. For each type of knowledge (i.e., association rules, clustering, classification, etc.), the components should be developed properly. Gupta et al. (2000a) identified five necessary components to achieve the complete functionality of the pattern I-MIN:

1. **Merge component:** Serial knowledge concentrated for windows (WKC) can be fused to create a wider temporary window. The merge operator provides the capacity to vary the window size to carry out the mining. It also establishes the equivalence between the accumulative knowledge concentrated (CKC) and the WKC. The existence of this component is fundamental for the support of WKCs. This operator is defined together with the design of the KC.  
The user decides the entrance to this component. The selected WKCs are fused and made available facts for the mining.
2. **Accumulation component:** It is the compound function that develops the premining functions, together with the aggregation and partial analysis of the data. The premining function is a compound function consisting of the selection, cleaning, and transformation operations. These functions are user-defined by means of the outline. However, the aggregation function is fixed by virtue of the data mining algorithm used.
3. **Mining component:** It is the main component of the pattern I-MIN. The mining query in the user’s interface invokes this component. The data structure used by the mining algorithm influences the design of the KC. It is important to emphasize that not all the mining algorithms are apt for intension mining. Algorithms, which inherently support incremental mining, are possible candidates. The mining algorithms, which fit in intension mining, should use data structures, which allow aggregation in some intermediate stage.
4. **Experimentation component:** This component is useful for analysis and interactive exploration of

data. It supports exploration centered on the user in an interactive way, by defining a group of operators. Each of these operators designates an experiment, which can be repeated using different perspectives. The following are three useful perspectives that have been recognized for experimentation:

- **Windowing:** to repeat the experiments exploring, selectively, different subsets of the database, varying the size of the windows dynamically.
  - **Variation of parameters:** to repeat the experiments with different metric or parameter values.
  - **Variation of restrictions:** to repeat the experiments with different restrictions. This functionality is supported by the iterative nature of the pattern I-MIN, with no need to carry out readings on the data repository. The operators process the KCs and evaluate the needed characteristics.
5. **Monitoring component:** This component has great potential in the detection of change patterns and their changes. Monitoring can be manual or automatic, according to the user's needs. Automatic monitoring can be specified in the schema at the designing time, or even later. A schema can define one or more monitoring instances. For the automatic monitoring, the support for triggers is essential. Manual monitoring is also possible by means of defined operators for experimentation. A friendly interface is required for this purpose, with a capacity to formulate easily comprehensible queries. Because the monitoring activity has associated aspects of privacy, only authorized users will be able to use this component.

## FUTURE TRENDS

The future in this area is fundamentally guided by the incorporation of some more data mining techniques, mainly those of classification and clustering. A potential area of expansion is fraud detection via system logs and Meta data (Gupta et al., 2000b).

The architecture permits knowledge discovery in platform and domain independent manner, knowledge preservation, knowledge renewal, and knowledge sharing for effective knowledge management (Gupta et al., 2001).

## CONCLUSION

Intension mining systems are much more efficient than the existing approaches and, as the database grows, effi-

ciency increases. It neatly integrates all the steps of KDP, as well (Gupta et al., 2000d).

The main benefits of intension mining are reducing search space and database scans, integration with databases, and clean approach as well as user centric mining, reuse of computation, exploration at two levels, support for active data mining, and flexibly exploration. The most important application of intension mining is electronic auditing (Gupta et al., 2000b).

Intension mining provides good interaction with the user. The possibility of users to generate queries and their periodic realization of the accumulation phase allow them to use exploration and monitoring as tools in knowledge discovery (Gupta et al., 2000c).

We think it is possible and of great advantage to integrate the intension mining framework with the Data warehouse.

## REFERENCES

- Agrawal, R., Kiernan, J., Srikant, R., & Xu, Y. (2002). Hippocratic databases. *Proceedings of the 28th Very Large Database Conference*. Retrieved March 2004 from [http://www.almaden.ibm.com/software/dm/Hippocratic\\_Databases/hippocratic.pdf](http://www.almaden.ibm.com/software/dm/Hippocratic_Databases/hippocratic.pdf)
- Bailey-Kellogg, C., & Ramakrishnan, N. (2003). Active data mining of correspondence for qualitative assessment of scientific computations. *Proceedings of the 17th International Workshop on Qualitative Reasoning*. Retrieved April 2004 from <http://www.cs.purdue.edu/homes/cbk/papers/qr03.pdf>
- Bhatnagar, V. (2001). *Intension mining: A new approach to knowledge discovery in databases*. Doctoral dissertation, Jamia Millia Islamia, New Delhi, India.
- Elmasri, & Narvathe. (2000). *Fundamentals of database system* (3<sup>rd</sup> ed.). Addison-Wesley Longman.
- Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., & Uthurusamy, R. (1996). *Advances in knowledge discovery and data mining*. Menlo Park, CA: AAAI Press.
- Ganti, V., Gehrke, J., & Ramakrishnan, R. (2000). DEMON-data evolution and monitoring. *Proceedings of ICDE*, San Diego, California.
- Goel, D. (1999). *A prototype for intension mining system*. Master's thesis, Indian Institute of Technology, New Delhi, India.
- Gupta, S. (2000). *Intension mining*. Slide presentation in the Second Workshop Mining Scientific Datasets. Retrieved November 2003 from the University of Minnesota



Web site: <http://www.cse.iitd.ernet.in/~skg/ppt/ahprc.ppt>

Gupta S., Bhatnagar, V., & Wasan, S. (2000a). *Intension mining: A new paradigm in knowledge discovery* (Tech. Rep. No. IITD/CSE/TR2000/001). Indian Institute of Technology, Department of Computer Science and Engineering. Retrieved November 2003, from <http://www.cse.iitd.ernet.in/~skg/ps/tr.ps>

Gupta, S., Bhatnagar, V., & Wasan, S. (2000b). Intension mining: A novel paradigm in KDD and D M. *Second Workshop in Mining Scientific Datasets*, University of Minnesota, Minneapolis.

Gupta, S., Bhatnagar, V., & Wasan, S. K. (2000c). User centric mining of association rules. *Workshop PKDD*, France. Retrieved November 2003, from <http://www.cse.iitd.ernet.in/~skg/ps/ddmi.ps>

Gupta, S., Bhatnagar, V., & Wasan S. K. (2000d). A proposal for data mining management system. *Workshop on Integrating Data Mining and Knowledge Management*, California. Retrieved December, 2003, from <http://cui.unige.ch/~hilario/icdm-01/DM-KM-Final/>

Gupta, S., Bhatnagar, V., & Wasan, S. (2005). Architecture for knowledge discovery and knowledge management. In *Knowledge and Information Systems*, 7(3), 310-336. London: Springer-Verlag.

Han, J., & Kamber, M. (2001). *Data mining: Concepts and techniques*. Morgan Kaufmann.

Imielinski, T., & Mannila, H. (1996). A database perspective on knowledge discovery. *Communications of the ACM*, 58-64.

Meo, R., Psaila, G., & Ceri, S. (1998). A tightly coupled architecture for data mining. *Proceedings of 14<sup>th</sup> International Conference on Data Engineering*, Orlando, Florida, USA, February, 23-27 (pp. 316-323).

Virmani, A. (1998). *Second generation data mining: Concepts and implementation*. Doctoral dissertation, Rutgers University, New Brunswick, NJ.

## KEY TERMS

**Active Data Mining:** Active data mining is concerned with the problem of integrating data collection, experiment design, and data mining with the end goal of making better use of data for data mining purposes. In applications where there is control over the data acquisition process, samples are preferred from locations that present the greatest benefit to identify high-level structures and

models underlying the data. Active data mining is especially crucial in sparse data contexts, in which each data point is costly (e.g., in terms of time, computational effort) and it is beneficial to make careful choices of locations to sample (Bailey-Kellogg & Ramakrishnan, 2003).

**CKC:** *Accumulative knowledge concentrate* condenses the complete history of the database. It is obtained by adding the increment of the database after overcoming the time limit. The structure of data created in the moment of the compilation of KDS continues growing. This type of KC is appropriate for applications purely oriented to mining or for limited experimentation. For questions related to monitoring, the dimension of time should be preserved (Gupta et al., 2000a).

**Hippocratic Databases:** Inspired by the Hippocratic oath, which states "Above all, do no harm." A system in which "contracts" are created between databases and the administrators—primary users. Thus, they ensure the privacy and integrity of data (Agrawal, Kiernan, Srikant, & Yu, 2002).

**Incremental Mining:** In most real-life applications, the data are incrementally accumulated in the database, and not completely altered overnight. This makes it possible to use the previously mined rules for updates, rather than reforming new ones.

**KC:** *Knowledge concentrates* capture the content of the information of the database in an instant. Additionally, they provide a window of the whole database for a certain period of time. An attaché of knowledge is represented by a structure of data derived from scanning the incremental database that can be used with mining purposes without having to request the database again. They are the outcomes of the accumulation phase, and they are specific for specified algorithm mining. Therefore, because KCs will constitute the entrance of the mining algorithm, their structure is closely bound to them. Finally, the sequences of KCs can so be mined, either for experimentation or monitoring purposes. There are two different types of KCs: *accumulative KC* (CKC) and *KC for windows* (WKC; Gupta et al., 2000a).

**KDA:** *The knowledge discovery administrator*. Similar to database administrators. their functions are to define the running time of the premining tasks in case the activity of the database is minimum, to centralize control, to incorporate new mining applications, and accounting (Gupta, 2000).

**KDD:** *Knowledge discovery in databases*. The non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data (Fayyad et al., 1996).



## **Intension Mining**

**KDP:** *Knowledge discovery process* is interactive and iterative. The development steps are understanding of the domain, defining the discovery goals, selecting relevant data, cleaning the data, transforming the data, mining the data, presenting the discovered knowledge, interpretation, and application (Han & Kamber, 2001). Some authors call the whole process *data mining*.

**KDS:** *Knowledge discovery schema* contains specification of premining requirements (i.e., data cleaning,

transformation, treatment of nulls, etc.), focusing, choice of algorithm (if possible or applicable), periodicity of accumulation, and choice of presentation tools (if possible or applicable; Gupta, 2000).

**WKC:** *Knowledge concentrated for windows*. The WKC capture the concentrated format of knowledge for a certain interval of time. In accordance with the specified frequency in the system, the WKC are generated periodically from the incremental database (Gupta et al., 2000a).

# Kernelized Database Systems Security

**Ramzi A. Haraty**

*Lebanese American University, Lebanon*

## INTRODUCTION

There are two main types of security in database systems: discretionary security and mandatory security. Discretionary security restricts access to data items at the discretion of the owner. Most commercial database management systems (DBMS) employ some form of discretionary security by controlling access privileges and modes of data users (Griffiths & Wade, 1976). Discretionary security is not adequate in a multilevel secure environment, however, because it does not prevent Trojan horse attacks and provides a low level of assurance. Mandatory security, on the other hand, restricts access of data items to cleared database users. It is widely exploited in military applications and provides a high assurance.

Numerous commercial and military applications require a multilevel secure database management system (MLS/DBMS). In a MLS/DBMS, database users are assigned classification levels and data items are assigned sensitivity levels. Usually, three architectures are used to provide security in MLS/DBMS. These architectures are the integrity lock or spray paint architecture, the data distribution architecture, and the kernelized architecture. The integrity lock and the data distribution architectures are outside the scope of this work. We focus only on the kernelized architecture.

In the kernelized architecture, data are physically stored in separate databases or containers according to sensitivity level. A multilevel relation is thus divided into single-level relations, each stored in a separate container. All containers are under the control of the common DBMS. The security of the system is largely dependent on the security of the operating system. Hence, the DBMS security is only as good as the underlying operating system. However, it is the responsibility of the MLS/DBMS to ensure that users can access only those data items for which they have been granted clearance.

The advantages of this architecture are that it is fairly secure and the kernel needed to implement it is relatively small. However, there are also several disadvantages. One major disadvantage is performance overhead associated with managing and executing multilevel transactions. In this article, we present an efficient model for concurrency control in kernelized databases. We show that the model is correct, secure, and provides a solution to the concurrency control problem.

## BACKGROUND

The standard military security approach is followed, which consists of two components: a set of security classes and a set of nonhierarchical compartments. The security classes are totally ordered from the lowest to the highest as follows: unclassified << confidential << secret << top secret. Within each security class there can be zero or more compartments (i.e., conventional, chemical, and nuclear).

A security class,  $S_1$ , is dominated by another class,  $S_2$  if  $S_2$  is hierarchically higher than  $S_1$  and contains all of its components.

In security terms, users or the processes that execute on behalf of them are referred to as subjects. Users are trusted, but processes are not. Objects, on the other hand, correspond to data items. Objects can be files, records, or even fields.

The Bell-LaPadula model introduced two security policies commonly accepted in a system that enforces multilevel security (Bell & LaPadula, 1976):

1. **Simple security policy:** A subject is allowed read access to an object if the subject's classification level is identical or higher than that of the object's sensitivity level.
2. **\*-policy:** A subject is allowed write access to an object if the subject's classification level is identical or lower than that of the object's sensitivity level.

These restrictions guarantee that proper access to objects will not be violated directly. However, they are insufficient to guarantee that proper access to objects is not violated indirectly through covert channels. Covert channels are channels, which are not intended to route information, but nevertheless do (Department of Defense, 1985). There are two main types of covert channels: covert storage channels and covert timing channels. Covert storage channels disclose information through the manipulation of a physical object, which either can or cannot be seen by low subjects. Covert timing channels can covertly send information from high to low subjects by modulating observable delay in the accessing of a common resource. A system is free from covert channels is called covert channel secure (CCS).

## MLS-ROLL

The simple security policy and a restricted version of the \*-policy of the Bell-LaPadula model were employed. The restricted version will allow a subject to have write access to an object if, and only if, the subject's classification level is equal to the object's sensitivity level (note that this is not a limiting restriction). The lattice-based model of information flow is followed, as proposed by Denning (1982), with some modifications.

Our proposed multilevel security request order linked list model (MLS-ROLL) is based on the ROLL concurrency control object (Perrizo, 1991). MLS-ROLL consists of two layers of linked list objects. The first is the global ROLL (GROLL) layer, and the second is the local ROLL (LROLL) layer, one at each security level.

When a high transaction requests a read access to a low data item, it must not set any lock on it; otherwise covert channels could be established. On the other hand, having such a transaction wait creates problems with unavailability. To eliminate these two problems, multiversioning of data is used. In our model, each write on a data item  $x$  produces a new version that follows the POST order of the transaction that wrote  $x$ . In this article, the term *version* refers to a value written by a committed transaction. To control storage cost, versions must periodically be purged. Because certain versions may be needed by active transactions, removal of older versions must be synchronized with respect to active transactions.

In our model, all write operations are "local" and some read operations are "global" (across security levels); hence, we call a transaction local or global depending on the types of operations it requests. When a local transaction,  $T_i$ , enters the system for execution, it is routed to the LROLL at the security level, which matches that of transaction  $T_i$  bypassing the GROLL completely. On the other hand, if  $T_i$  is a global transaction, a global request vector (GRV) is created in which one bit is set for each container for which access is required. The GRV is POSTed to the GROLL. Using the CHECK operation on the GROLL,  $T_i$  determines to which container it can locally POST. After successful POSTing, the local container sends an acknowledgment message to the GROLL. GROLL can then RELEASE the corresponding bit for that particular container in the GVR in GROLL.

If  $T_i$  requires read access to data item  $x$ , such that the security level of  $T_i$  is greater than or equal to the security level of  $x$ , that is,  $SL(T_i) \geq SL(x)$ , then the transaction manager that is running on behalf of  $T_i$  denoted by  $TM_i$ , CHECKs the ROLL at  $SL(x)$  to find the POST order of the last transaction that is ahead of  $T_i$ 's entry point value and that also has a write lock on  $x$ . If

there is such a transaction, say  $T_j$ ,  $TM_i$  waits until  $T_j$  commits and then reads the highest version of  $x$ . Otherwise,  $T_i$  reads the highest version of  $x$  that is less than or equal to  $T_i$ 's entry point.

All reads that are at the same level as the transaction are also performed the same way, except that a transaction compares its own POST order, instead of entry point, with that of the other transactions that are ahead of it. On the other hand, the write operations can be executed, creating new versions of data items, without performing any CHECKs for conflicting operations.

We now present MLS/ROLL formally:

1. There is a GROLL that all global transactions must POST to. GROLL is responsible for maintaining consistency of transaction POSTings across containers.
2. There is a separate ROLL object at each security level.
3. When a local transaction,  $T_i$ , enters the system for execution, its request vector (RV) will be POSTed in the ROLL object that exists at the same level as  $T_i$ .
4. When a global transaction,  $T_j$ , enters the system for execution, a GRV is created in which one bit is set for each local container where access is needed.
5. The GROLL is CHECKed periodically to see if any of the containers requested by  $T_j$  is available.
  - A. If yes, then  $T_j$  is sent for local POSTing
  - B. If no, GROLL is reCHECKed again.
- 5.1 If the container available is at the same level as  $T_j$ ,  $TM_j$  POSTs  $T_j$ 's RV in the ROLL object that is at the same container.
- 5.2 If the container available is at lower level, then  $TM_j$  gets the entry point (i.e., the POST order of the last transaction in the ROLL).
6. When a transaction,  $T_j$ , is POSTed locally, its corresponding bit is RELEASEd in the GROLL. The process continues until all the bits in the GRV of  $T_j$  are RELEASEd.
7. When transaction  $T_j$  wants to read a data item  $x$  such that  $SL(x) < SL(T_j)$ , then  $TM_j$  checks the ROLL at  $SL(x)$  from its entry point onwards to check for any conflicts:
  - 7.1 If a conflict exists, then  $TM_i$  waits until there are no more conflicts. Then  $T_j$  reads the highest version of  $x$ .
  - 7.2 If there is no conflict, then  $T_j$  reads the highest version of  $x$  that is less than or equal to its entry point.

8. When  $T_j$  wants to read  $x$  such that  $SL(x) = SL(T_j)$ ; then  $TM_j$  CHECKS for any conflicting transactions that are ahead of it in the ROLL at the same level.  $TM_j$  reads  $x$  in a way that was described in steps 7.1 and 7.2.
9. When transaction  $T_j$  wants to write a data item  $x$ , it does so without CHECKING for any conflicting operations in the ROLL and thus produces a new version of  $x$ .

## CORRECTNESS OF MLS-ROLL

- **Theorem:** The multiversion history (H) produced by MLS-ROLL in one-copy serializable. In fact, H is equivalent to a one-serial history in which transactions are placed in the POST order.
- **Proof:** To prove that the multiversion serialization graph MVSG(H) is acyclic, we first show that every edge in the MVSG(H) follows the POST order of transactions (Bernstein, Hadzilacos, & Goodman, 1987). That is, if  $T_i \rightarrow T_j$  is in MVSG(H), then  $PO(T_j) \rightarrow PO(T_i)$  where  $PO(T_i)$  and  $PO(T_j)$  denote the post order of transactions  $T_i$  and  $T_j$  respectively.

As a transaction POSTs only once in MLS-ROLL, and the POST partial orders across containers is the same for the same set of transactions, there would never be a cycle in the post order. Therefore, there would never be a cycle in MVSG(H).

## COMPARISON WITH OTHER SOLUTIONS

The area of multilevel security in database systems is fairly new. However, many algorithms have been proposed so far. Haraty (1999) proposed a security policy manager for object-oriented paradigms. Getta (2003) proposed a hybrid scheduler that scrambles messages sent from high level transactions with a random binary sequence. Son, David, and Thuraisingham, (1996) considered elimination of covert channels by introducing a secure two-phase locking scheduler. Lee, Jeong, and Seung (1996) proposed an improved version. Binto and Haritsa (1997) presented different schedulers to resolve inter- and intra-level conflicts among transactions. Keefe, Tsai, and Srivastava (1990) proposed a scheduler, which deals with multiversioning of data and a priority queue of transactions based on their classification levels. Hence, a

high transaction will always be scheduled before any active low transactions in the system, thus reading stale information. In addition, this algorithm requires the scheduler to be trusted. Jajodia and Atluri (1991) overcame these problems by using an improved timestamp-ordering algorithm. In his algorithm, a high transaction, when rejected for reading a low level data item that later was modified by another low level transaction with a lesser timestamp, the former transaction reexecutes starting from the same data item. Consequently, all data items read by that transaction up to that point are not re-read. Thus, the transaction encounters the possibility of reading stale data. Although this algorithm does not introduce any signaling channels and eliminates livelocks between transactions at different classification levels, it does not rule out the possibility of starvation of a transaction at its own level. For example, if a transaction  $T_i$  wants to write a data item  $x$ , the scheduler selects a version  $xk$  with the largest timestamp,  $wts(xk)$ , such that  $wts(xk) < ts(T_i)$ , where  $ts(T_i)$  denotes the timestamp of  $T_i$ .  $T_i$  would be rejected if the read timestamp of  $xk$ ,  $rts(xk) > ts(T_i)$ . Then  $T_i$  would be resubmitted with a newer timestamp, but still there is the possibility for rejection after each resubmission because of the aforementioned reason. In our algorithm, transactions do not suffer from livelocks because ROLL itself is livelock free. Moreover, all high level transactions read the current version of data. It also produces serializable executions.

## FUTURE TRENDS

Future work will involve taking a closer look MLS/DBMS and define new and better ways of handling transaction management, as well as query processing. Future work will also involve extending security issues to temporal and multimedia databases.

## CONCLUSION

In this article, a model was presented that I hope will solve the concurrency control problem existing in MLS/DBMSs. Security-related issues were discussed, and MLS-ROLL was presented, and I proved that MLS-ROLL is correct. Informal comparison with other published solutions were also presented. Future work will include extending multilevel security to multidatabase environments.

## REFERENCES

- Bell, D., & LaPadula, L. (1976). *Secure computer systems: Unified exposition and multics interpretation* (Tech. Rep. MTR 2997). Bedford, MA: MITRE Corp.
- Bernstein, P., Hadzilacos, V., & Goodman, N. (1987). *Concurrency control and recovery in database systems*. Boston: Addison-Wesley.
- Binto, G., & Haritsa, J. (1997). Secure transaction processing in firm real-time database systems. *Proceedings of the ACM SIGMOD Conference*, Tuscon, Arizona.
- Denning, D. (1982). *Cryptography and data security*. Boston: Addison-Wesley.
- Department of Defense. (1985). *Department of defense trusted computer system evaluation criteria*. National Computer Security Center.
- Getta, J. (2003). Hybrid concurrency control in multi-level secure database systems. *Proceedings of the 21<sup>st</sup> IASTED International Conference – Applied Informatics*, Innsbruck, Austria.
- Griffiths, P., & Wade, B. (1976). An authorization mechanism for a relational database system. *ACM Transactions on Database Systems*, 1(3), 242-255.
- Haraty, R. (1999). A security policy manager for multi-level secure object oriented database management systems. *Proceedings of the International Conference on Applied Modeling and Simulation*, Cairns, Queensland, Australia.
- Jajodia, S., & Atluri, V. (1991). Alternative correctness criteria for concurrent execution of transactions in multilevel secure databases. *Proceedings of the IEEE Symposium on Research in Security and Privacy*. Oakland, CA.
- Keefe, T., Tsai, W., & Srivastava, J. (1990). Multilevel secure database concurrency control. *Proceedings of the 6<sup>th</sup> International Conference on Data Engineering*, Los Angeles, CA.

Lee, S., Jeong, B., & Seung, H. (1999). A secure dynamic copy protocol in real-time secure database systems. *Proceedings of the ACM SIGPLAN Workshop on Languages, Compilers, and Tools for Embedded Systems* (pp. 73-79).

Perizzo, W. (1991). Request order linked list (ROLL): A concurrency control object for centralized and distributed database systems. *Proceedings of the 7<sup>th</sup> International Conference on Data Engineering*, Kobe, Japan.

Son, A., David, R., & Thuraisingham, B. (1996). Improving timeliness in real-time secure database systems. *SIGMOD Record*, 25(1), 29-33.

## KEY TERMS

**Deadlock:** A situation in which a transaction in a set of transactions is blocked while waiting for another transaction in the set, and therefore none will become unblocked (unless there is external intervention).

**Discretionary Security:** A form of security where access to data items is restricted at the discretion of the owner.

**Kernelized Architecture:** An architecture in which data are physically stored in databases according to sensitivity level.

**Livelock:** A situation in which a transaction is continually aborted and restarted but is never given the opportunity to terminate normally.

**Mandatory Security:** A form of security in which access to data items is restricted to cleared database users.

**Multilevel Secure Database Management Systems:** A system whereby database users are assigned classification levels and data items are assigned sensitivity levels.

**Serializable:** Produces the same output and has the same effect as that of a serial execution.



# Knowledge Discovery and Geographical Databases

**Sami Faïz**

*National Institute of Applied Sciences and Technology, Tunisia*

## INTRODUCTION

Geographic data are characterized by huge volumes, lack of standards, multiplicity of data sources, multi-scale requirements, and variability in time. These characteristics make geographic information complex and uncertain. At the same time, the important growth of the quantity of data manipulated and the necessity to make rapid decisions imposed the appearance and the great progress of new tools like data warehousing techniques. Data warehouse is usually defined as a subject-oriented, integrated, time-variant, and non-volatile collection of data in support of decision-making processes.

Several techniques have been coined since the appearance of data warehousing: OLAP (On Line Analytical Processing) and data mining techniques.

## BACKGROUND

Due to processing time and the important increase of geographic data (Faïz, 1999), data warehouses have appeared recently to answer to the new requirements of the database technologies. Along with these data warehouses appeared some automated techniques like OLAP and data mining, which have become in great demand in relational, transactional, and geographical databases because of their complexity.

Data warehouses have been defined in several ways. Some authors say that it is an instantiated view of some integrated information to build mediators, which are views of many integrated information sources.

Several techniques have been coined since the appearance of data warehousing:

- Data mining techniques refer to the extraction of implicit, previously unknown, and potentially useful information from large databases. Spatial data mining deals with extracting knowledge from spatial databases.
- OLAP: a particularity of these tools is to systematically preaggregate data, which favor rapid access to information. Therefore, the storing structure is always cubic or pyramid-like. Various algorithms per-

mitting the manipulation of cubes of data has been proposed.

## MAIN THRUST OF THE ARTICLE

We are going to introduce the principles of spatial data warehouses, which are used in the management of a large volume of geographic data to provide a foundation for analytical processing. Then, we introduce the tools permitting exploration of the spatial data warehouses; spatial data mining and spatial OLAP (Faïz, 2000).

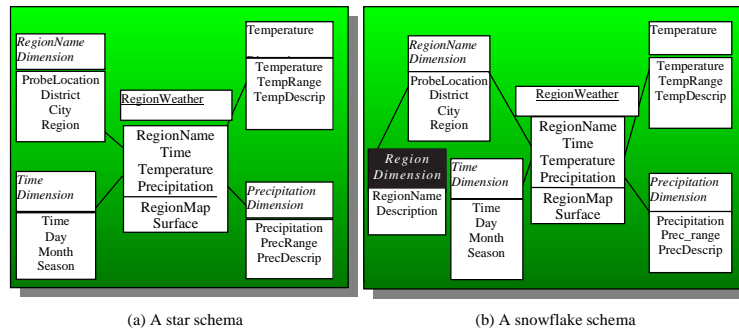
### Spatial Data Warehouses

In a spatial data warehouse, data can be combined from many heterogeneous sources to obtain decision support tools. These sources have to be adjusted because they contain data in different representations (Bâazaoui, Faïz & Ben Ghezala, 2000). Therefore, the construction of a data warehouse requires many operations, such as integration, cleaning, and consolidation.

To build a spatial data warehouse, there are two fundamental models. The star schema model is usually used to describe and represent the multi-dimensional data model (Agrawal, Gupta & Sarawagi, 1997; Golfarelli, Maio & Rizzi, 1998). Figure 1(a) shows an example of star schema representation; the database contains a central fact table (RegionWeather Table) and a number of dimension tables (RegionName Dimension, Temperature Dimension, Precipitation Dimension, etc.). The fact table is the only table with multiple joins connecting to other tables, and the other dimension tables have only a single join attaching them to the fact table. Each tuple in the fact table consists of a pointer to each of the dimension tables that provide their multi-dimensional coordinates, and stores the numeric measurements for those coordinates. Each dimension table consists of columns that correspond to attributes of the dimension.

The other model used is Snowflake schema as shown in Figure 1(b). It provides a refinement of star schemas where dimensional hierarchies are explicitly represented by normalizing the dimension tables. This leads to advantages in maintaining the dimension\_tables. How-

Figure 1. The two fundamental models to build a data warehouse



ever, the denormalized structure of the dimensional tables in star schemas may be more appropriate for browsing the dimensions. Data warehouses also store select summary tables containing preaggregated data, which correspond generally to aggregating the fact table on one or more selected dimensions.

### Spatial Data Mining and Spatial OLAP

Spatial data mining is an extraordinarily demanding field referring to extraction of implicit knowledge and spatial relationships, which are not explicitly stored into geographical databases (Han, Koperski & Stefanovic, 1997). In fact, the collected data can exceed human capacity to analyze and extract interesting knowledge from large spatial databases (Faiz, Abbassi & Boursier, 1998).

The most popular method of knowledge discovery in geographic databases consists in the discovering of spatial association rule. Some defined this method, which consists in finding interesting association or correlation among a large set of data, to identify sets of attributes called predicates or items that frequently occur together.

### Examples

- $home(x) \wedge near(x, sea) \rightarrow expensive(x)$   
(90%) (70%)
- $great\_station(x) \rightarrow near(x, high\_way)$   
(80%) (90%)

The form of association rule:

$$P1 \wedge \dots \wedge Pm \rightarrow Q1 \wedge \dots \wedge Qn \quad (C\%) \quad (P\%)$$

Where C% is called the confidence of the rule. It is the percentage of the conclusion (Q) when condition (P) is true. P% is called the support of the rule. It is the

percentage of transactions that contain the condition (P) among the whole of the transactions.

Many spatial predicates can be presented in this kind of rule: (1) topology relation disjoint, intersect, inside, outside, adjacent\_to; (2) spatial orientations right, left, north, south; and (3) distance information near\_to, far\_from, and so on.

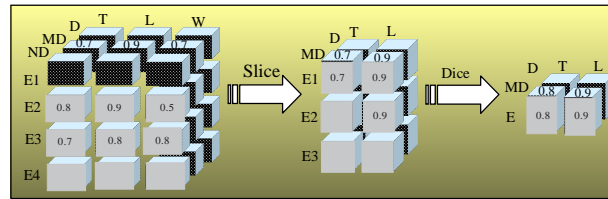
Spatial OLAP (SOLAP) can be defined as a visual platform built especially to support rapid and easy spatio-temporal analysis and exploration of data following a multi-dimensional approach comprised of aggregation levels available in cartographic displays as well as in tabular and diagram displays (Bédard, 2002; Rivest, Bédard & Marchand, 2001). The spatial OLAP technology uses multi-dimensional views of aggregated, pre-packaged, and structured data to have quick access to information. This technology facilitates queries on large quantities of data. Aggregated data must be archived in cubic or pyramid-like structures (Abello, Samos & Saltor, 2001).

Spatial OLAP essentially includes three steps:

1. Selecting data from spatial data warehouse
2. Building data cube
3. Online analysis using the cube

The data structure of a data cube consists of dimensions, which are the edges of the cube and values in each cell called measures. Each dimension refers to dimension table, which describes it. For example, a dimension table for Region may contain attributes description, area, and so forth that can be identified by the data analysts. A number of operations on data cube may manipulate dimensions and multiple levels of abstraction: roll-up (increasing the level of abstraction), drill-down (decreasing the level of abstraction or increasing detail), slice and dice (selection and projection), pivot (reorienting the multi-dimensional view of data), and so on.

Figure 2. The use of operations: Slice & Dice



By the association of the cartographic representation and OLAP navigation, the user moves in the multi-dimensional structure and obtains representations of the data via cartographic, tabular posting or statistical diagram which are a function of dimensions, measurements, and the selected levels of hierarchy.

### EXAMPLE

One of the main objectives of GIS combined with data warehouse tools is to help decision makers in making the best decision when handling natural or human resources. Then, the quality of the decision heavily depends on the quality of geographic data.

A GIS user needs to be confident in the result of the analysis. The user has to be sure of the status of data in terms of errors (absence or presence of data, wrong coding, imprecise location). The knowledge of the quality of data before any use helps the user to fix the framework. Geographic data control consists in processing some automatic or semiautomatic operations on data. These operations will help in finding and measuring the different kinds of errors. We need estimators to represent the result of measurements: bias on X and Y, standard deviation on X and Y, true, false, missing ratio, present ratio, and so on. The question then concerns the organization and storage of this quality information (Bâazaoui Faïz & Ben Ghezala, 2003).

In this example, we show that the cubic representation permits a good managing of quality information. The result of geographic data control is the generation of quality information that users will take into account during data processing and, more specifically, during the decision process.

The fundamental question is the following: “How will we show the quality information?” In our cube, we have three dimensions:

- **Entity-dimension:** we consider the set of roads (E1, E2, E3...).

- **Attribute-dimension:** we consider three attributes (Thickness of the road, Length of the road, and Width of the road).
- **Density-dimension:** each entity belongs to a (Dense, Non Dense and Medium Dense) zone.

We can notice in this cube that the value 0.9 refers to the quality information of E2 length attribute, which belongs to the non-dense zone. Consequently, this cube gives information on the quality of each road’s attribute. We can extract from this cube some measurements of quality, like average-quality of the width attribute for the whole roads, which belong to a medium-density, or a pondered average-quality. With these dimensions, OLAP operations will permit moving up and down along any dimension shown in the cube, for example, Slicing and Dicing: each one selects portion of the cube based on the constant(s) in one or few dimensions. For instance, one may be interested only in E1, E2, and E3 entities, which are not non-dense. They are characterized by thickness and length attributes, as shown in Figure 2.

### FUTURE TRENDS

Due to the complex characteristics of geographic databases (huge volumes, lack of standards, multiplicity of data sources, multi-scale requirements, and variability in time), spatial data warehouse, spatial data mining, and spatial OLAP techniques have appeared. Therefore, the construction of the spatial data warehouse is not easy. Our perspective is to work on the modeling and implementation of spatial data warehouse by using a meta model.

### CONCLUSION

In this article, we have already introduced data warehouses and given an overview of the most important

tools of data warehousing: OLAP and data mining. The necessity of these automated methods is created by the huge amount of manipulated data applied to geographical databases. The data mining techniques permits the discovery of non-trivial information, while OLAP prepares aggregations of information from the databases to rapidly answer the queries of the user. At the end, we have presented an example to demonstrate how geographical quality information can be managed by using a multi-dimensional presentation of data (cube).

## REFERENCES

- Abello, A., Samos, J., & Saltor, F. (2001). Understanding analysis dimensions in a multidimensional object-oriented model. *Proceedings of the 3rd International Workshop on Design and Management of Data Warehouses (DMDW'01)*. Switzerland.
- Agrawal, R., Gupta, A., & Sarawagi, A. (1997). *Modeling multidimensional databases (ICDE'97)*, (pp. 232-243).
- Bâazaoui, H., Faïz, S., & Ben Ghezala, H. (2000). Geographical datawarehouses and quality. *Proceedings of the International Conference on Artificial and Computational Intelligence for Decision, Control and Automation in Engineering and Industrial Applications (ACIDCA'2000)*, Monastir, Tunisia.
- Bâazaoui, H., Faïz, S., & Ben Ghezala, H. (2003). CASME: A case tool for spatial data marts design and generation. *Proceedings of the 5th International Workshop on Design and Management of Data Warehouses (DMDW'03) in conjunction with the 29th International Conference on Very Large Databases (VLDB'03)*, Berlin, Germany.
- Bédard, Y. (2002). *Geospatial data warehousing, datamart and SOLAP for geographic knowledge discovery*. University of Muenster, Germany.
- CNIG. (1993). *Quality of exchanged geographical data*. Report from the National Council of Geographical Information, Paris, France.
- Faïz, S. (1999). *Geographic information system: Quality information and data mining*. Editions C.L.E.
- Faïz, S. (2000). Managing geographic data quality during spatial data mining and spatial OLAP. *International Journal of GIM*, 28-31.
- Faïz, S., Abbassi, K., & Boursier, P. (1998). *Applying datamining techniques to generate quality information within geographical databases*. In Jeansoulin & Goodchild (Eds.), *Data quality in GI*. Paris: Editions Hermès.
- Golfarelli, M., Maio, D., & Rizzi, S. (1999). The dimensional fact model: A conceptual model for data warehouses. *International Journal of Cooperative Information Systems*.
- Han, J., Koperski, K., & Stefanovic, N. (1997). GeoMiner: A system prototype for spatial data mining. *Proceedings of the ACM-SIGMOD Conference on Management and Data (SIGMOD'1997)*, Tucson, Arizona (pp. 553-556).
- Rivest, S., Bédard, Y., & Marchand, P. (2001). Towards better support for spatial decision-making: Defining the characters ppatial on-line analytical processing (SOLAP) *Geomatica: The Journal of the Canadian Institute of Geomatics*.

## KEY TERMS

**Data Mining:** Known as knowledge discovery from databases and refers to the process of extracting interesting and non-trivial patterns or knowledge from databases. Spatial data mining is an extraordinarily demanding field referring to extraction of implicit knowledge and spatial relationships, which are not explicitly stored into geographical databases.

**Data Warehouse:** An instantiated view of integrated information sources to build mediators. In a spatial data warehouse, data can be combined from many heterogeneous sources to obtain decision support tools. These sources have to be adjusted because they contain data in different representations. Therefore, the construction of a data warehouse requires many operations such as integration, cleaning, and consolidation.

**Geographic Data:** Characterized by the fact that they are constituted of two kinds of attributes: descriptive or non-spatial attributes, and positioning or spatial attributes. Non-spatial data are not specific of geographic applications, and they are usually handled by standard relational DBMSs. Spatial attributes need additional power in order to design, store, and manipulate the spatial part of geographic entities.

**GIS:** Geographic Information System (GIS) is a computer system capable of capturing, storing, analyzing, and displaying geographically referenced information.

**OLAP:** On-Line Analytical Processing. A particularity of these tools is to systematically preaggregate data, which favor rapid access to information. Therefore, the storing structure is always cubic or pyramid-like. Various algorithms permit the manipulation of cubes of data. Spatial OLAP can be defined as a visual platform built especially to support rapid and easy spatio-temporal analysis and exploration of data following a multidimensional approach comprised of aggregation levels available in cartographic displays as well as in tabular and diagram displays.

**Spatial Data Mining:** Refers to the process of extracting interesting and implicit knowledge and spatial relationships, which are not explicitly stored into geographical databases.

**Spatial OLAP:** A visual platform built especially to support rapid and easy spatio-temporal analysis and exploration of data following a multidimensional approach.



# Knowledge Discovery from Databases

**Jose Hernandez-Orallo**

*Technical University of Valencia, Spain*

## INTRODUCTION

As databases are pervading every parcel of reality, they record what happens in and around organizations all over the world. Databases store the detailed history of organizations, institutions, governments, and individuals. An efficient and agile analysis of the data recorded in a database can no longer be done manually.

Knowledge discovery from databases (KDD) is a collection of technologies that aim at extracting non-trivial, implicit, previously unknown, and potentially useful information (Fayyad, Piatetsky-Shapiro, Smyth, & Uthurusamy, 1996) from raw data stored in databases. The extracted patterns, models, or trends can be used to better understand the data, and hence, the context of an organization, and to predict future behaviors in this context that could improve decision making. KDD can be used to answer questions such as, Is there a group of customers buying a special kind of product? Which sequence of financial products improves the chance of contracting a mortgage? Which telephone call patterns suggest a future *churn*? Are there relevant associations between risk factors in coronary diseases? How can I assess if my e-mail messages are more or less likely to be *spam* (junk mail)?

The previous questions cannot be answered by other tools usually associated with database technology, such as OLAP tools, decision support systems, executive information systems, and so forth. The key difference is that KDD does not convert information into (more aggregated or interwoven) information, but generates inductive models (under the form of rules, equations, or other kinds of knowledge) that could be sufficiently consistent with the data. In other words, KDD is not a deductive process but an inductive one.

## BACKGROUND

The more and more rapid evolution of the context of organizations forces the revision of their knowledge relentlessly (what used to work before no longer works). This provisional character of knowledge helps explain why knowledge discovery from databases, still in its inception, is so successful. The models and patterns that can be obtained by data mining methods are useful for

virtually any area dealing with information: finance, insurance, banking, commerce, marketing, industry, private and public healthcare, medicine, bioengineering, telecommunications, and many other areas (M. J. A. Berry & Linoff, 2004).

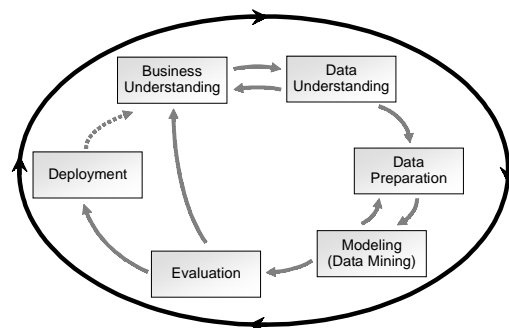
The name KDD dates back to the early 1990s and is not a new “technology” itself; KDD is a heterogeneous area that integrates many techniques from several different fields without prejudices, incorporating tools from statistics, machine learning, databases, decision support systems, data visualization, World Wide Web research, among others, in order to obtain novel, valid, and intelligible patterns from data (Berthold & Hand 2002; Dunham 2003; Han & Kamber, 2001; Hand, Mannila, & Smyth, 2000).

## PROCESS OF KNOWLEDGE DISCOVERY FROM DATABASES

The KDD process is a complex, elaborate process that comprises several stages (Dunham, 2003; Han & Kamber, 2001): data preparation (including data integration, selection, cleansing, and transformation), data mining, model evaluation and deployment (including model interpretation, use, dissemination and monitoring).

Cross-industry standard process for data mining (CRISP-DM; see <http://www.crisp-dm.org> for more information) is a standard reference that serves as a guide to carry out a knowledge discovery project and comprises all the stages mentioned (see Figure 1).

*Figure 1. Stages of the knowledge discovery process*



According to this process, data mining is just a stage of the knowledge discovery process. The data mining stage, also called the *modeling stage*, converts the prepared data (usually in the form of a “minable view” with an assigned task) into one or more models. This stage is thereby the most characteristic of the whole process, and data mining is frequently used as a synonym for all the process. The process is cyclic because, after a complete cycle, the goals can be revised or extended to start the whole process again.

Many references to KDD choose the data preparation stage as the start of the knowledge discovery process (Dunham, 2003; Fayyad, 1996; Han & Kamber, 2001). The CRISP-DM standard, however, emphasizes that business understanding and data understanding (which includes data integration) must be handled before data preparation. This is reasonable because the first thing to know before starting with a KDD project is what the business needs are, to establish the business goals according to the business context, to see whether there is (or if we can get) enough data to solve them, and, in this case, to specify the data mining objectives.

For example, consider a distribution company that has a problem of inadequate stocks, which generate higher costs and frequently make some perishable products expire. From this problem, one business objective could be to *reduce the stock level of perishables*. If there is an internal database with enough information about the orders performed by each customer in the last 3 years, and we can complement this with external

information about the reference market sale prices of these product categories, it could be sufficient to start looking for models to address the business objective. This business objective could be translated into one or more data mining objectives (e.g., predict how many perishable products by category the customer is to buy week by week, from the orders performed during the last three years and the current market sale price).

Before starting the discussion about each specific KDD stage, it is important to highlight that many key issues on the success of a KDD project have to do with a good understanding of the goals and the feasibility of these goals, as well as a precise assessment of the resources needed (e.g., data, human, software, organizational). Table 1 shows the most important issues for success in a KDD project.

Additionally, a major reason for a possible failure in KDD projects may stem from an excessive focus on technology, that is, implementing data mining because others do, without recognizing what the needs of the organization are and without understanding the resources and data necessary to cover them.

## Data Integration and Preparation

Once the data mining goals are clear and the data that will be required to achieve them is located, it is necessary to get all the data and integrate them. Usually, the data can come from many sources, either internal or external. The typical internal data, and generally the main source of

Table 1. Keys to success in a knowledge discovery project

- Business needs must drive the knowledge discovery project. The business problems and, hence, the **business objectives must be clearly stated**. These business objectives will help one understand the data that will be needed and the scope of the project.
  - Business objectives must be translated into specific **data mining objectives**, which must be relevant for the organization. The data mining objectives must be accompanied with a specification on the required quality of the models to be extracted in terms of a set of metrics and features: expected error, reliability, costs, relevance, comprehensibility, and so forth.
  - The knowledge discovery project must be integrated with other plans in the organization and must have the unconditional **support from the organization executives**.
  - **Data quality** is crucial. The integration of data from several sources (internal or external), its neatness and an adequate organization and availability (using a data warehouse, if necessary) is a sine qua non in knowledge discovery.
  - The **use of integrated and friendly tools** is also decisive, helping the process on many issues, not only for the specific stages of the process but also on other issues, such as documentation, communication tools, workflow tools, and so on.
  - The **need of a heterogeneous team**, comprising not only professionals with a specific data mining training but also professionals from statistics, databases and business. A strong leadership among the group is also essential.
- To translate the positive results of knowledge discovery into positive results for the organization, it is necessary to use a **holistic evaluation and an ambitious deployment of the models** to the know-how and daily operation of the organization.

information, include one or more transactional databases, possibly complemented with additional internal data, in the form of spreadsheets or other kinds of formatted or unformatted data. External data include all other data that can be useful to achieve the data mining objectives, such as demographic, geographic, climate, industrial, calendar, or institutional information.

Some small-scale data mining projects can be done on a small set of tables or even on a couple of data files. But, generally, the size, different sources, and formats of all this information require the integration into one single repository. Repositories of this kind are usually called data warehouses. Data warehouses and related technologies such as OLAP are covered by another entry in this encyclopedia. The reader is referred to this entry for more information.

It is important to note, however, that the data requirements for a data warehouse for OLAP analysis are not exactly the same that those for a data warehouse whose purpose is data mining. Data mining usually requires more fine-grain data (less aggregation) and must gather the data needed for the data mining objectives, not for performing complex reports or complex analytical queries. Even so, OLAP tools and data mining tools can work well together on the same data warehouse, which is designed taking into account both needs.

Along or after the task of determining the required data and integrating them, there is an even more thorny undertaking: data cleansing, selection, and transformation. The quality of data is even more important for data mining than it is for transactional databases. Nevertheless, because the data used for data mining are generally historical, we cannot act on the sources of the data. In many situations, the only possible solution is *cleansing* the data, by a heterogeneous set of techniques grouped around the term *data cleansing*, which might include exploratory data analysis, data visualization, and other techniques.

Data cleansing mostly focuses on *inconsistent* and *missing* data. First, *inconsistent data* can only be detected by comparing the data with database constraints or context knowledge or, what is more interesting, by using some kind of statistical analysis for detecting *outliers*. Outliers might or might not be errors, but they at least get the attention of an expert or tool to check whether they are, indeed, errors. First, in this case, there are many possibilities, including removing the entire row or the entire column, substituting the value by a null, a mean, or an estimated value, among others. Second, *missing data* may seem easier to detect, at least when the data come from databases that use null values, but can be more difficult if the data repository comes from several sources that are not using null values to represent unknown or nonapplicable data. The ways to handle missing values are

analogous to those for handling wrong data. Data cleansing is essential because many data mining algorithms are highly sensitive to *missing* or *erroneous* values.

Having an integrated and clean data repository is not sufficient to perform data mining on it. Data mining tools may not be able to handle the *overall* data repository, or this would be simply prohibitive. Consequently, it is important to select the relevant data from the data repository, denormalize and transform them, in order to set up a view, which is called the *minable view*. Data selection can be done horizontally (data sampling or instance selection) or vertically (feature selection). Data transformation includes a more varied collection of operations, including the creation of derived attributes (by aggregation, combination, numerization, or discretization), by principal component analysis, de-normalization, pivoting, and so forth.

Finally, data integration and preparation must be done with a certain degree of understanding about the data itself, and frequently requires more than half of the time and resources of the whole knowledge discovery process. Furthermore, working on the integration and preparation of data, and especially through data visualization (Fayyad, Grinstein, & Wierse, 2001), generates a better understanding of the data, and this might suggest new data mining objectives or could show that some of them are unfeasible.

## Data Mining

The input of the data mining, modeling, or learning stage, is usually a *minable view* with a task. A data mining task specifies the kind of problem to be solved and the kind of model to be obtained. Table 2 shows the most important data mining tasks.

Data mining tasks can be divided into two main groups: predictive and descriptive, depending on the use of the patterns obtained. Predictive models predict future values of unseen data, such as, *a model that predicts the sales for next year*. Descriptive models describe or aim to understand the data, such as, *there is a strong sequential association between contracting the free-mum-calls service and churn*.

To solve a data mining task, we need to use a certain data mining technique, such as decision trees, neural networks, linear models, support vector machines, and so forth. A list of the most popular data mining techniques is shown in Table 3 (Berthold & Hand, 2002; Han & Kamber, 2001; Hand, Mannila, & Smyth, 2000; Mitchell, 1997; Witten & Frank, 1999). There are some techniques that can be used for several tasks. For instance, decision trees are generally used for classification but can also be used regression and for clustering. Other techniques are specific for solving one single task.

Table 2. Data mining tasks

- Predictive
  - *Classification*. The model predicts a nominal output value (one from two or more categories) with one or more input variables. Related tasks are categorization, ranking, preference learning, class probability, and estimation.
  - *Regression*. The model predicts a numerical output value with one or more input variables. Related tasks are sequential prediction and interpolation.
- Descriptive
  - *Clustering*. The model detects “natural” groups in the data. A related task is summarization.
  - *Correlation and factorial analysis*. The relation between two (bivariate) or more (multivariate) numerical variables is ascertained.
  - *Association discovery (frequent itemsets)*. The relation between two or more nominal variables is determined. Associations can be undirected, directed, or sequential.

Table 3. Data mining techniques

- Exploratory data analysis and other descriptive statistical techniques
- Parametrical and nonparametrical statistical modeling (e.g., linear models, generalized linear models, discriminant analysis, Fisher linear discriminant functions, logistic regression)
- Frequent itemset techniques (e.g., Apriori algorithm and extensions, GRI)
- Bayesian techniques (e.g., Naïve Bayes, Bayesian networks, EM)
- Decision trees and rules (e.g., CART, ID3/C4.5/See5, CN2)
- Relational and structural methods (e.g., inductive logic programming, graph learning)
- Artificial neural networks (e.g., perceptron, multilayer perceptron with backpropagation, Radial Basis Functions)
- Support vector machines and other kernel-based methods (e.g., margin classifiers, soft margin classifiers)
- Evolutionary techniques, fuzzy logic approaches, and other soft computing methods (e.g. genetic algorithms, evolutionary programming, genetic programming, simulated annealing, Wang & Mendel algorithm)
- Distance-based methods and case-based methods (e.g., nearest neighbors, hierarchical clustering [minimum spanning trees], *k*-means, self-organizing maps, LVQ)

For instance, the a priori algorithm is specific for association rules.

The existence of so many techniques for solving a few tasks is because no technique can be better than the rest for all possible problems. Techniques also differ in other features: some of them are numerical, others can be expressed in the form of rules, some are quick, others are quite slow. Hence, it is useful to try an assortment of techniques for a particular problem and to retain the one that gives the best model.

### Model Evaluation, Interpretation, and Deployment

Any data mining technique generates a tentative model, a hypothesis, which must be assessed before even thinking about using it. Furthermore, if we use several samples

of the same data or use several algorithms for solving the same task, we will have several models, and we have to choose from them. There are many evaluation techniques and metrics. Metrics usually give a value of the *validity*, *predictability*, or *reliability* of the model, in terms of the expectation of some defined error or cost. The appropriate metrics depend on the data mining task. For instance, a classification model can be evaluated with several metrics, such as *accuracy*, *cost*, *precision/recall*, *area under the ROC (receiver operating characteristic) curve*, *log loss*, and many others. A regression model can be evaluated by other metrics, such as *squared error*, *absolute error*, and others. Association rules are usually evaluated by *confidence/support*.

To estimate the appropriate metric with reliability it is well known that the same data that was used for training should not be used for evaluation. Hence, there are several techniques to do this better (e.g., *train and test evaluation*, *cross-validation*, *bootstrapping*). Both the evaluation metric and the evaluation technique are necessary to avoid *overfitting*, a typical problem of learned models, (i.e., *lack of generality*).

Once the model is evaluated as feasible, it is important to interpret and analyze its consequences. Is it comprehensible? Is it novel? Is it consistent with our previous knowledge? Is it useful? Can be put into practice? All these questions are related to the pristine goal of KDD, to obtain nontrivial, previously unknown, comprehensible, and potentially useful knowledge.

Next, if the model is novel and useful, we would like to apply or scatter it inside the organization. This deployment can be done manually or by embedding the obtained model into software applications. For this purpose, standards for exporting and importing data mining models, such as the predictive model markup language (PMML) standard, defined by the Data Mining Group (see <http://www.dmg.org> for more information), can be of great help here.

Last but not least, we must recall that the knowledge is always provisional. Therefore, we must monitor the models with the new incoming data and knowledge and revise or regenerate them when they are no longer valid.

### FUTURE TRENDS

Current and future areas of research are manifold. First, KDD can be specialized depending on the kind of data handled. Data mining has not only be applied to structured data in the form of, usually relational, databases, but it can also be applied to other heterogeneous kinds of data (e.g., *semistructured data*, *text*, *Web*, *multimedia*, *geographical*). This has given names to the following specific areas of KDD: *Web mining* (Kosala & Blockeel,



2000), text mining (Berry, 2003), and multimedia mining (Simoff, Djeraba, & Zaïane, 2002). Intensive research is being done in these areas.

Other, more general areas of research include dealing with specific, heterogeneous or multirelational data, the latter known as relational data mining (Dzeroski & Lavrac, 2001), data mining query languages, data mining standards, comprehensibility, scalability, tool automation and user friendliness, integration with data warehouses, and data preparation (Domingos & Hulten, 2001; Roddick, 1998; Srikant, 2002; Smyth, 2001). The progress in all these areas will make KDD an even more ubiquitous and indispensable database technology than it is today.

## CONCLUSION

This article has shown the main features of the process of KDD: goals, stages and techniques. KDD can be considered inside the group of other analytical, decision-support, and business intelligence tools that use the information contained in databases to support strategic decisions, such as OLAP, EIS, DSS or more classical reporting tools. However, we have shown that the most distinctive trait of KDD is its inductive character, which converts information into models, data into knowledge.

There are many other important issues around KDD not discussed so far. The appearance of data mining software constituted the final boost for KDD. During the 1990s, many companies and organizations released specific data mining algorithms and general purpose tools, known as *data mining suites*. Although, initially, the tools focused on integrating an assortment of data mining techniques, such as decision trees, neural networks, logistic regression, and so on, nowadays the suites incorporate techniques for all the stages of the KDD process, including techniques for connecting to external data sources, data preparation tools, and data evaluation and interpretation tools. Additionally, data mining suites, either vendor-specific or general, are being more and more integrated with DBMS and other decision support tools, and this trend will be reinforced in the future.

## REFERENCES

- Berry, M. J. A., & Linoff, G. S. (2004). *Data mining techniques: For marketing, sales, and customer relationship management* (2nd Ed.). New York: Wiley.
- Berry, M. W. (2003). *Survey of text mining: Clustering, classification, and retrieval*. New York: Springer-Verlag.
- Berthold, M., & Hand, D. J. (Eds.). (2002). *Intelligent data analysis. An introduction* (2nd Ed.). Berlin: Springer.
- Domingos, P., & Hulten, G. (2001). *Catching up with the data: Research issues in mining data streams*. Workshop on Research Issues in Data Mining and Knowledge Discovery.
- Dunham, M. H. (2003). *Data mining. Introductory and advanced topics*. Upper Saddle River, NJ: Prentice Hall.
- Dzeroski, S., & Lavrac, N. (2001). *Relational data mining*. Berlin: Springer.
- Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., & Uthurusamy, R. (1996). *Advances in knowledge discovery and data mining*, Cambridge, MA: MIT Press.
- Fayyad, U. M., Grinstein, G., & Wierse, A. (2001). *Information visualization in data mining and knowledge discovery*. San Francisco, CA: Morgan Kaufmann.
- Han, J., & Kamber, M. (2001). *Data mining: Concepts and techniques*. San Francisco, CA: Morgan Kaufmann.
- Hand, D. J., Mannila, H., & Smyth, P. (2000). *Principles of data mining*. Cambridge, MA: MIT Press.
- Kosala, R., & Blockeel, H. (2000). Web mining research: A survey. *ACM SIGKDD Explorations*, Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining, 2, 1-15.
- Mitchell, T. M. (1997). *Machine learning*. New York: McGraw-Hill.
- Roddick, H. (1998). Data warehousing and data mining: Are we working on the right things? In Y. Kambayashi, D. K. Lee, E.-P. Lim, Y. Masunaga, & M. Mohania (Eds.), *Advances in database technologies: Lecture notes in computer science, 1552* (pp. 141-144). Berlin, Germany: Springer-Verlag. Lecture Notes in Computer Science. 1552.
- Simoff, S. J., Djeraba, C., & Zaïane, O. R. (2002). MDM/KDD 2002: Multimedia data mining between promises and problems. *SIGKDD Explorations*, 4(2), 118-121.
- Smyth, P. (2001). *Breaking out of the Black-box: Research challenges in data mining*. Workshop on Research Issues in Data Mining and Knowledge Discovery.
- Srikant, R. (2002). *New directions in data mining*. Workshop on Research Issues in Data Mining and Knowledge Discovery.
- Witten, I. H., & Frank, E. (1999). *Tools for data mining*. San Francisco: Morgan Kaufmann.



## KEY TERMS

**Association Rule:** A rule showing the association between two or more nominal attributes. Associations can be directed or undirected. For instance, a rule of the form, *If the customer buys French fries and hamburgers she or he buys ketchup*, is a directed association rule. The techniques for learning association rules are specific, and many of them, such as the a priori algorithm, are based on the idea of finding frequent item sets in the data.

**Classification Model:** A pattern or set of patterns that allows a new instance to be mapped to one or more classes. Classification models (also known as classifiers) are learned from data in which a special attribute is selected as the “class.” For instance, a model that classifies customers between likely to sign a mortgage and customers unlikely to do so, is a classification model. Classification models can be learned by many different techniques: decision trees, neural networks, support vector machines, linear and nonlinear discriminants, nearest neighbors, logistic models, Bayesian, fuzzy, genetic techniques, and so forth.

**Clustering Model:** A pattern or set of patterns that allows examples to be separated into groups. All attributes are treated equally and no attribute is selected as “output.” The goal is to find “clusters” such that elements in the same cluster are similar between them but are different to elements of other clusters. For instance, a model that groups employees according to several features is a clustering model. Clustering models can be learned by many different techniques: *k*-means, minimum spanning trees (dendrograms), neural networks, Bayesian, fuzzy, genetic techniques, and so forth.

**Data Mining:** It is a central stage in the process of knowledge discovering from databases. This stage transforms data, usually in the form of a “minable view” into some kind of model (e.g., decision tree, neural network, linear or nonlinear equation). Because of its central place in the KDD process and its catchy name, it is frequently used as a synonym for the whole KDD process.

**Machine Learning:** A discipline in computer science, generally considered a subpart of artificial intelligence, which develops paradigms and techniques for making computers learn autonomously. There are several types of learning: inductive, abductive, and by analogy. Data mining integrates many techniques from *inductive* learning, devoted to learn general models from data.

**Minable View:** This term refers to the view constructed from the data repository that is passed to the data mining algorithm. This minable view must include all the relevant features for constructing the model.

**Overfitting:** A frequent phenomenon associated to learning, wherein models do not generalize sufficiently. A model can be overfitted to the training data and perform badly on fresh data. This means that the model has internalized not only the regularities (patterns) but also the irregularities of the training data (e.g., noise), which are useless for future data.

**Regression Model:** A pattern or set of patterns that allows a new instance to be mapped to one numerical value. Regression models are learned from data where a special attribute is selected as the *output* or *dependent* value. For instance, a model that predicts the sales of the forthcoming year from the sales of the preceding years is a regression model. Regression models can be learned by many different techniques: linear regression, local linear regression, parametric and nonparametric regression, neural networks, and so forth.

# Knowledge Management in Tourism

**Daniel Xodo**

*Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina*

**Héctor Oscar Nigro**

*Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina*

## INTRODUCTION

Management greatly depends on knowledge, and its detection, creation, transmission, and number of intangibles play a fundamental role in success.

In tourism, information systems must work on immaterial concepts and take steps to satisfy the expectations of multiple potential customers. These systems require complex models of reality and suitable conceptual tools to work out strategies.

This article expounds the use of different mathematical and management techniques that can be applied to the modeling, application, and control of strategic and operative management.

## BACKGROUND

### Data, Information, and Knowledge

These three concepts, which are frequently used as synonyms, are different representation stages of reality and its apprehension enables us in different ways.

*Data*, as simple representations, do not tell us much about themselves unless they are related to other data. It is from this relation that we get *information*.

Information is useful because of the possibility of intervention (e.g., cause-effect relationships), which is provided by *knowledge*.

What is knowledge? Knowledge can be understood as *representation*, *production* or *estate* (Marakas, 1999). In relation to this definition, any of the aforementioned concepts are, generically, *knowledge*. The definition of *knowledge* will depend on the usage and application of the term.

If we consider knowledge as “behavior support” (Lopez, 1999)—sharing perceptions in a socioeconomic system by means of the knowledge of goals, means, and evaluation of accomplishments—it will be easier to accomplish it effectively and efficiently.

If we consider knowledge *taxonomically*, we could derive, by means of appropriate processes, a linguistic and assimilative level of conceptualization and shared

understanding from a primary level (descriptive, procedural, and rational). Hence, the possibilities of organization management success will be improved. Concepts such as “mission” are hard to explain, as its vagueness facilitates a certain *mission* to be attractive to many people who, at the same time, give their own meaning to it that does not always coincide with that of the others. The same happens with the comprehension of different functions, qualities, or behaviors, which must be shared by numerous actors.

The information systems meant to communicate knowledge vary according to what we want to transmit. However, it is always necessary to reduce ambiguity and unify concepts.

The following four types of *formal systems* can be distinguished (Simon, 1995):

- a. of beliefs
- b. of limit setting
- c. of diagnosis control
- d. of interactive control

In any of these, knowledge can be *represented*, *generated*, or *established* in different states, according to its usage, relationship, and application. For example, a certain number of guests in a hotel may be rather meaningless, but if we add the capacity, the date, and the category of the hotel to the number of people, it will have different meanings for the owner, the competitor or a new guest. These meanings will be influenced by, among other factors, the vision, the knowledge, and the expectations of the people involved. In this way, we will build up representation models, whose complexity will favor or damage our decisions according to our ability to deal with them.

### Knowledge as Representation: The Indicators

If we take the concept “intellectual capital” to mean *useful knowledge*, which we develop “increasing the human capital” (Olve, Roy, & Wetter, 2000), it is obvious that people’s knowledge varies from individual to

individual. It is necessary to “translate the vision and the strategy into goals and indicators, through a well-balanced set of perspectives” (Kaplan & Norton, 1999). The indicators will give the necessary homogeneity to the perceptions of those who have to make decisions in relation to the reality given by the indicator as well as its interpretation (i.e., people’s subjective judgment).

### **Knowledge as Production**

The occurrence of events, their measuring by means of indicators, and their processing provide elements of judgments, references, parameters, and interpretation of the reality on which decisions are made. In the dynamic of measuring and evaluating reality, we will “produce” new knowledge that will modify the evaluation as well as our behavior. A clear example is to study the environmental impact produced by tourist activity through intensity measures, and its extension and importance over the socioeconomical system (Coccosis & Parpairis, 1992).

### **Knowledge as State**

According to its relevance, knowledge can be perceived in different ways: data, information, structured information, perceptions, judgments, and decisions (van Louizen, 1996). We are constantly changing the knowledge from one level to another until we get to the culmination of the knowledge, the *decision*. It is in this process that the precision and meaning of the information increases

At the same time, in the same process, we must reconcile *material* factors and goals with *cultural* and *subjective* factors (e.g., information, computing projects planning, information management with system of thoughts, learning systems, activity systems; Espinoza & Molina, 1999).

### **Knowledge Management in Tourism**

Considering knowledge management as the process of managing immaterial assets explicitly, we immediately come to know that management in tourism (either private or public) is an especially fertile field. The aims of the service are mostly experiences, individual or group acquired knowledge, emotions, and personal transformations (Gunn, 1994). Likewise, the possibility of offering the service properly depends on the physical infrastructure and its usage as well as on the ability of interpreting expectations and needs, evaluating subjectivities and, getting ahead of them, understanding customs, likes and dislikes, preferences, and cultural and social guidelines (Tissen, Andriesen, & Lekanne,

2000) and from the offer of services and hiring until arrival time, staying, leaving, and even the memories and the will to come back (Organización Mundial de Turismo, 1997).

The challenge is to make the tourist destination offer the guests the desired utility, getting as a reward a successful operation of the system. To achieve this, the prodigal nature or the large investments are not enough. There are other elements that should be considered, such as proper management concepts directed towards constant learning and innovation, service processes, orientation to the customers, and the proper application of the economic–financial resources (Inskeep, 1991; Wisniewski & Dickson, 2001).

### **Service Profit Chain**

Among the diverse approaches and strategies to increase the quality of the provision facilities, the customers’ satisfaction, and, as a consequence, the increase in the incomes and the final profitability, it is useful to mention the “utility chain in services” (Heskett, Sasser, & Schlesinger, 1998).

The utility chain links different concepts that have an effect on the improvement of services, related to the ability of the ones whose mission is to carry them out. Almost all of them are related to knowledge, individual and shared knowledge of

- a. structures
- b. processes
- c. needs
- d. motivations for choice
- e. service complementarity
- f. demand complementarity
- g. development of special packages for different segments
- h. customers satisfaction
- i. vision
- j. personnel loyalty and satisfaction
- k. investments in relevant aspects

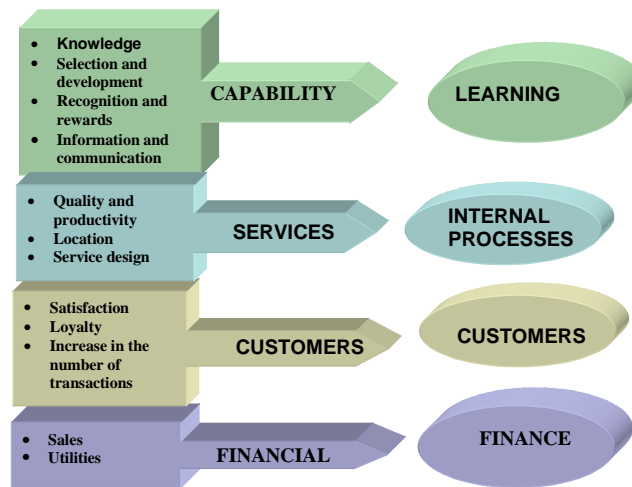
These concepts can be appreciated in the “value equation” (see equation below).

$$\text{Value equation} = \frac{\text{Results} + \text{Process quality}}{\text{Price} + \text{Cost of access to a service}}$$

The keys for its application would be in

- a. understanding the client needs
- b. responding or covering needs

Figure 1. Service profit chain and balanced scorecard



- c. investing in key aspects, such as clients' restrain
- d. developing value packages for different markets
- e. developing value conscience (Heskett et al.,1998; Ho & McKay,2002)

The service profit chain and its correspondence to the balanced scorecard is shown in Figure 1.

The service profit chain (Heskett et al., 1998) shows the relationship between different fundamental factors of tertiary activities such as tourism. This shows that the successful results in a certain administration come out as a result of the employees' capability, loyalty, satisfaction and productivity; generating high value services due to quality and low costs. Hence, not only do clients' satisfaction and loyalty rise, but so do growth and utilities.

Higher value is bounded to major commitment and people's training (growing perspectives). As a result, there will be better performed processes (internal process perspective) with more effective results (customer perspective), and thus better financial results (financial perspective). There is an important correspondence between the utility chain and the balanced scorecard perspectives.

### Balanced Scorecard as a Tool of Knowledge Management

We can focus on knowledge in relation to two principal theories: information or people management (Probst, Raub, & Romhardt, 2000).

In information management, all the information is included at its different levels and support systems. In people management, its equivalence in the complex dynamic of human behavior takes particular significance. Both concepts should be considered within the frame of the *structure* in which they are developed or used. Therefore, we have three different elements at our disposal: information, people, and structure.

It is at this stage when the balanced scorecard functions as a knowledge management tool: the choice, diffusion, and application of the chosen strategy in the selected reference system. The data the BSC put together are useful to those people who are responsible for the application and control of the strategy. For example, the hotel reservation rate is useful to foresee the ticket sales to a certain tourist destination. In addition, the rate will be the result of the advertising and service strategies chosen and it will allow service providers to measure its efficiency (Miyake, 2002).

### Need for Knowledge in Tourism

The basic knowledge in the sector are based on

- a. physical assets
- b. knowledge processes in tourist services (intangible assets)
- c. strategies:
  - choice
  - diffusion
  - application

This knowledge is detected, elaborated, and transmitted through ratios, outcome measures and performance drivers whose substantial value is to reflect the cause-effect relationship of the items they are linking (Baud-Bovy, 1998; McIntyre,1992).

### Balanced Scorecard as a Means for Managing Knowledge

The balanced scorecard studies different aspects in the knowledge management of an organization and combines cultural, structural, and strategic aspects by causality relationships through

- a. mission and vision
- b. strategies
- c. responsibilities
- d. application

- modeling of tasks
  - modeling of performance
- e. transformation

### Balanced Scorecard Advantages with Regard to Knowledge Management

- a. it considers intangible assets
- b. it makes use of linked *outcome measures* and future performance drivers
- c. it makes the mission and the strategy more comprehensive by means of action
- d. it is a cyclical process
- e. it considers intellectual factors: creativity, innovation
- f. its main utility is communication
- g. it is multidimensional
- h. it allows selecting indicators and inductors
- i. it transfers the strategic outcome to quantitative measures
- j. it transforms the strategy into action
- k. it quantifies qualitative variants
- l. it links cause–effect relationships to strategic activities
- m. it is prospective
- n. it allows measuring, simulating and evaluating alternatives
- o. it can be quickly modified (Kaplan & Norton, 2001; Malina & Selto, 2001; Neidorf, 2002)

The balanced scorecard (see Figure 2) outlines the tourist activity showing diverse cause–effect relationships.

### Indicators

Outcome measures and performance drivers. The detection and selection of the *indicators* and *inductors* must be taken into account as diverse aspects to materialize the balanced scorecard.

- a. cause–effect relationships
- b. the possibility of obtaining, storing, and updating information
- c. feasible control and processing
- d. facility for diffusion and interpretation

## FUTURE TRENDS

### Informatic Applications in Tourist Knowledge Management

#### Data Warehouse, Data Mining, and Online Analytical Processing (OLAP) Applications and Use

Applications of these techniques (Fayyad, 1996, 1997) include

- a. detection
- b. selection of detection tools
- c. integration of tools
- d. application–organization
- e. computerization and oversight

#### Data Warehouse as an Organization Process of the BSC

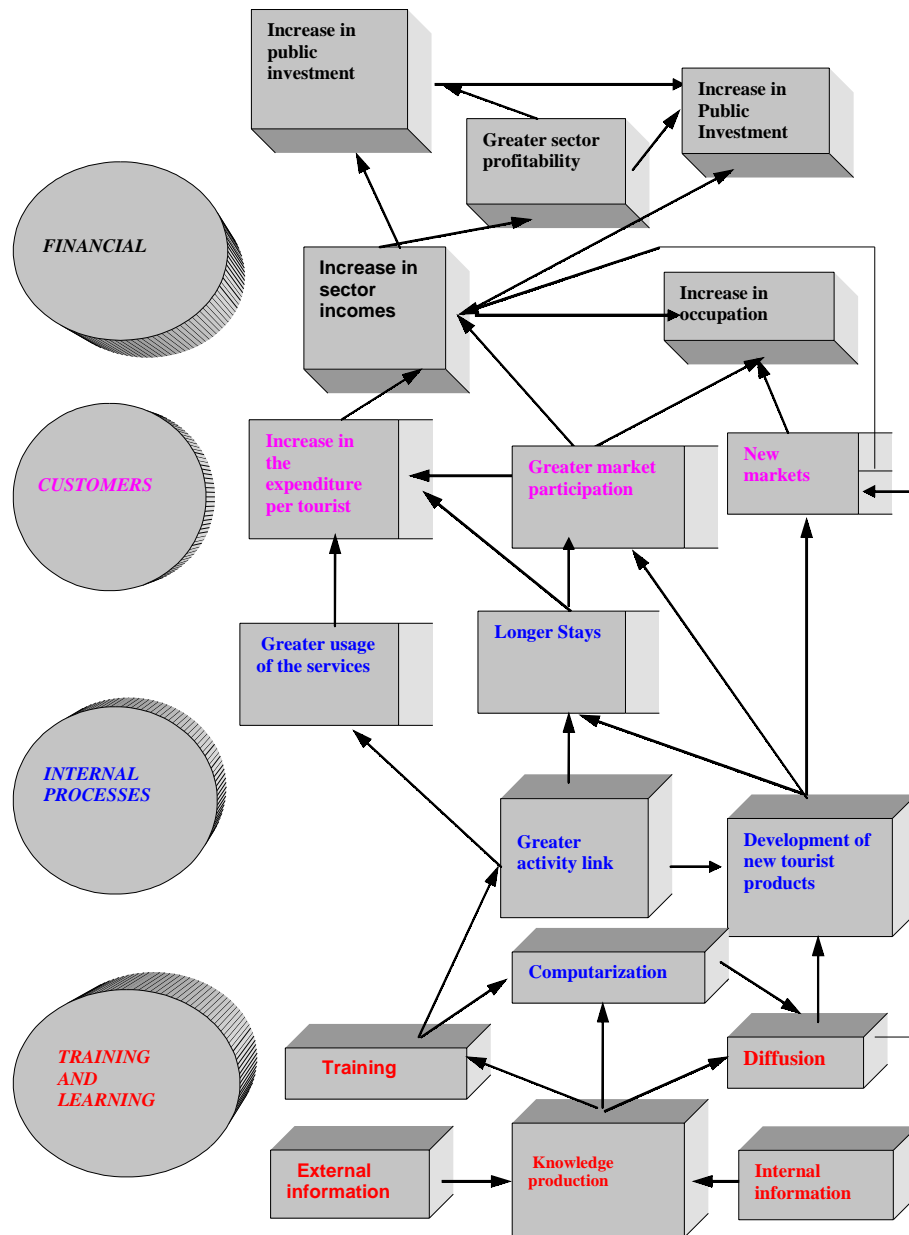
Data warehouse is a collection of data aimed at the matter—integrated, not volatile, thematic, and historical—organized to support a decision-making helping process. (Inmon, 1996).

#### Useful Data Warehouse Characteristics for a BS

- **Integrated:** The data stored in the data warehouse are integrated in a structure; therefore, any existent inconsistency among the diverse operational systems must be eliminated. The information is often arranged into different detail levels for its adjustment to the users' needs.
- **Thematic:** Only the data essential for the knowledge generation process is integrated from the operational environment. The data is organized by topics to facilitate their access and understanding to the final users. For example, customers' data can be compiled in a unique data warehouse board. Consequently, the requests of information about customers will be easier to answer.
- **Historical:** Time is an implicit part of the information compiled in a data warehouse. In operational systems, data usually show the state of business activity at present. The information stored in a data warehouse can be used, among other purposes, to do trend analysis.



Figure 2. Some cause-effect relationships in tourism



- **Nonvolatile:** The data of a data warehouse are meant to be read, no to be modified. Therefore, the information is permanent. Updating the data warehouse is the incorporation of the latest values the different variables it contains have taken without taking any kind of action on what already existed (Inmon, Glassey, & Welch, 1997).

### OLAP Tools for a Balanced Scorecard Information Analysis

We shall add the multidimensional information analysis by means of OLAP tools and consider OLAP systems as parts of the executive information systems (EIS), which are used to provide the strategic level with the necessary information for decision making (Codd, Codd, & Salley, 1993).



In an OLAP data model, the information is perceived as cubes that consist of *descriptive categories* (dimensions) and *quantitative values* (measures). The multidimensional data model makes it easier for the users to formulate complex queries, correct the mistakes in a report, change summed-up data for detailed data, and “filter” the data in meaningful subsets.

## Applying Data Mining Tools in a Balanced Scorecard

### Data Mining Definition

Data mining is an analytical process that has been designed to explore large quantities of data and to search for consistent models and the systematic relationships among variables so as to validate the results of applying the discovered models to the new data subsets.

The process consists of three steps: *exploration, construction or definition of the model, and validation/verification.*

If the nature of the available data allows, we repeat it interactively until we identify a “strong” model. However, in business practice, the possibilities to validate the model in the analysis phase are usually limited. Consequently, the initial results often have the heuristic condition which might influence in the decision-making process.

There are three main working areas: *knowledge engineering, classification, and problem solving.* Each learning technique places itself in this three-dimensional space. No learning or pattern recognition technique is considered the best. An environment of knowledge databases discovery must bare these different types of techniques (hybrid environment, Indurkha & Weiss, 1998).

## Different Learning Algorithms Compared to Different Types of Tasks in the Application Inside a BSC

Any of the three techniques (i.e., knowledge engineering, classification, and problem solving) can be applied in a balanced scorecard to analyze and predict indicators or to take part in the indicators’ building process (see Figure 3).

### Structure and Application to the Proposed Indicators

Neuronal Networks Application in the Prediction of an Indicator. If we take a resulting indicator as the *increase*

*in the percentage of the average stay*, we will get, in addition to this current indicator and its historical trajectory (i.e., applying temporal series), an indicator of the potential average stay for the following 6 months. The source of information of this indicator can be build up from (a) hotel and outing databases, and (b) sample interviews considering hotels and outings.

A temporal series represents the evolution of a magnitude in time. The factor of highest interest will be the correlation between different events along time. If such a correlation exists and can be modeled, predictions of the future behavior of the temporal series can be made.

The temporal series are build-up on a *feed-forward* architecture, such as multilayer perception (MLP) or radial basis functions (RBF). Alternatives include feed-forward networks, with a fixed-entry window and recurrent networks with unique entries.

There are two types of models that can be applied to predict the next value in a temporal series:

- single-step prediction, in which the inputs to the model are always known values taken from the temporal series, and
- multi-step prediction, in which the result of the first prediction is fed back as a new entry in the network (see Figure 4).

### Studying Techniques

There are two ways of studying temporal series:

- finding out patterns to explain the past behavior of the temporal series, and
- evaluating the effect of a fact that intervenes and changes the behavior of the temporal series.

Figure 3. Learning algorithms

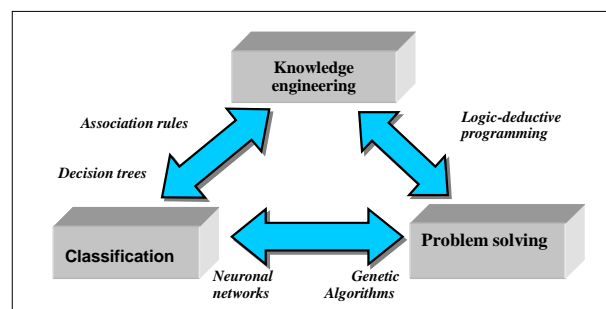
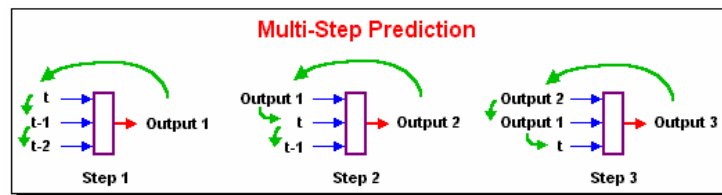


Figure 4. Sequence of steps in the prediction



**Cycle Development: Preproject → Data Gathering → Data Preparation → Design → Training and Testing → Implementation and Maintenance**

Between the design, and training and testing stages there is an interactive stage that corresponds to *optimization*.

**Preparation of Data**

The data employed in the application consisted of the following five fields registered (or calculated) in the T instant:

- a. An increase in the percentage of the average stay (result indicator)
- b. Quantity of necessary consultancies to define the trip
- c. Increase in the number of options of tourist packages
- d. Quantity of innovations in the Web site
- e. Information availability in service centers

The last four items are guiding indicators.

**Design**

Selection of neuronal processing tools, model structure, and initial conditions.

- a. Data input tool
- b. Time series window
- c. Time series plot

**Data Input Tool**

The data input tool allows the specifying of the variables to be included in the input window and also the ones that will go in the target window (prediction).

**Time Series Windows**

Time series windows allow the making of two kinds of predictions: single-step prediction and multi-step prediction.

**Time Series Plot**

Time series plot shows the results of an application as a two-dimensional graphic as a function of time. It shows the temporal comparison between the actual and predicted outputs and the actual outputs with inputs.

**INDICATORS GROUPING TO GENERATE ANOTHER GUIDING INDICATOR**

An example of this is taking into account the customers' level and Kohonen network.

The generated guiding indicator is "type of customer," with the following characteristics:

- a. A varied passengers' origin
- b. Percentage of reiteration of visits
- c. Percentage of provenances
- d. Segments diversity
- e. Number of visitors
- f. Total expenditure per tourist

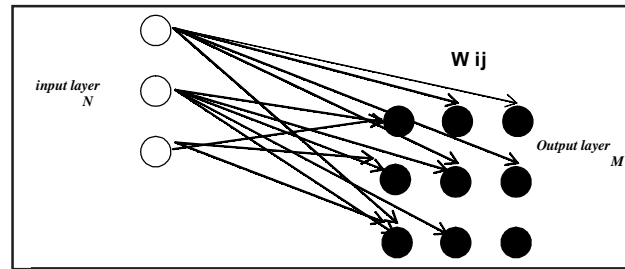
**Kohonen Auto-Organized Maps**

These are based on evidences found at brain level. They associate entry vectors with output patterns and are able to make cluster analysis (i.e., to project a high-dimensional space onto a smaller one).

The learning process is nonsupervised competitive. The neurons compete with each other to carry out a



Figure 5. SOFM architecture



given task. When there is just one input, only one neuron activates (i.e., the winning neuron).

- **Goal:** “Cluster” the data introduced to the network.

### SOFM Architecture: Connections

Each entry memory  $i$  is connected to each of the output neurons of  $j$  by means of weight ( $W_{ij}$ ). Each output neuron has a weight vector associated to it ( $W_{ij}$ ) as a reference vector, which represents the average vector of this category. Among the neurons of the output layer there are implicit lateral connections of excitement or inhibition (del Brío, 1997; see Figure 5).

### Learning Stage

This stage aims to establish, by means of the presentation of a training pattern set, the different categories that will be employed during the working stage to classify new input patterns. A set of patterns is presented repeatedly in the network, until the different reference vectors are tuned to one or more input patterns. If the input space is divided into groups, each neuron will specialize in one of them, and the essential operation of the network will be interpreted as a “cluster analysis” (see Figure 6).

### Working Stage

In the output layer, each neuron estimates the similarity between the input vector ( $X_p$ ) and its own weight vector ( $W_{ij}$ ). Simulating a competitive process, the winning neuron will be the one whose weight vector resembles the input vector. The activated output neuron represents the class the input belongs to. When the input is a similar pattern, the same neuron, or a neighbor neuron, is activated.

### Application of the Fuzzy Decision Theory as a Financial Indicator of the Future Investment in the Area

Fuzzy logic deals with imprecise information (such as the acceptance of a tourist package, the proper customers’ attention, or an uncomfortable means of transport to the city) in terms of fuzzy sets (Lazzari, Machado, & Pérez, 1998). These fuzzy sets are turned into rules to define actions, such as “if the quality of a service is not good, the amount of tourists will decrease 50% in the following 6 months.”

Control systems based on fuzzy logic combine input variables (which are defined in terms of fuzzy sets) by means of rules that produce one or several output values (Kasabov, 1996). The fuzzy set theory comes from the classic set theory and adds a belonging function to the set, which is defined as a real number between 0 and 1.

The concept of fuzzy sets or subsets is introduced in association with a determined linguistic value, defined by a word, adjective, or linguistic label  $A$ .

For each fuzzy set or subset there is a belonging or inclusive function  $f_a(t)$  that indicates the degree in which the  $t$  variable is included in the concept that label  $A$  represents (Hilera & Martínez, 1995). For example, the linguistic value *customers’ attention* may represent the acceptance degree in percentage to the attention of a specific tourist place expressed through interviews. Then, we can define three fuzzy sets, each one identified

Figure 6. Bidimensional map sample

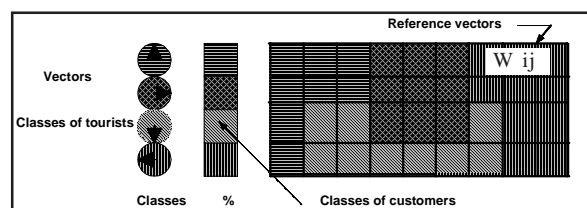
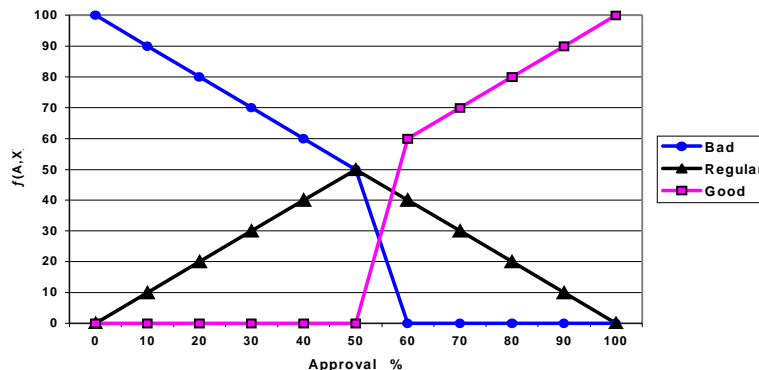


Figure 7. Fuzzy sets



with a label (*Bad*, *Regular*, or *Good*) and with a belonging or inclusive function.

$$\{f_{Bad}(t), f_{Regular}(t), f_{Good}(t)\}.$$

The outcome would be a distribution, as shown in Figure 7.

An economic–financial indicator to apply to the scorecard may be the amount of acquired packages for a specific tourist program. We will apply the fuzzy decision theory mechanisms to deduce different decisions about the acquisition of those packages and the future profits linked to each option (e.g., for each travel agent), according to the ways of searching information, the attitudes towards risk, the decision criteria, the total uncertainty, or others.

For these purposes, the concept of fuzzy logic may be applied, estimating the number of package tours to be sold, who will make decisions about the amount of reservations periodically booked in a specific excursion program. These decisions will have an initial capital investment intended to establish booking cost for each package to estimate the profitability.

## CONCLUSION

Management has powerful formal tools at its disposal to optimize decisions. The computational decision systems (e.g., management information systems, decision support systems, executive information systems) are samples of the growing trend to eliminate empiricism, to go deeply into the analysis, to release from duty those who decide on routines, thus allowing them to adopt approaches of greater conceptual amplitude. Among these approaches are the *ideals* of understanding, vi-

sion, and quality, whose scope broadens the deterministic and quantitative approaches. Within this frame, the methods of proximal reasoning (in which we tend to include as an essential element the typical uncertainty that the social sciences constitute), together with computational algorithms, is a promising path in search of knowledge, which is a human concern for all people at all times.

## REFERENCES

- Baud-Bovy, M. (1998). *Tourism and recreation: Handbook of planning and design*. Oxford, UK: Architectural Press.
- Bonsón, E. (1999). *Tecnologías inteligentes para la gestión empresarial*. Madrid, Spain: RA-MA Editorial.
- Coccosis, H., & Parpairis, A. (1992). *Tourism and the environment. Some observations on the concept of carrying capacity*. Dordrecht, The Netherlands: Kluwer Academic Press.
- Codd, F., Codd, B., & Salley, C. (1993). *Providing OLAP to user-analysts*. Codd.
- del Brío, M. S. A. (1997). *Redes neuronales y sistemas borrosos, introducción teórica y práctica*. Madrid, Spain: RA-MA Editorial.
- Espinoza E. M., & Molina S. C. (1999). Cambio organizacional: Sistemas de información y emociones. *Gestión y Estrategia*, 15.
- Fayyad, U., & Simoudis, E. (1997). *Data mining and knowledge discovery in data bases*. Cambridge, MA: MIT Press.





- Fayyad, U. M., & Piatetsky-Shapiro, G. (Eds.). (1996). *Advances in knowledge and data mining*. Cambridge, MA: MIT Press.
- Grunewald, L. (Ed.). (2001). *Plan de desarrollo integral de la actividad turística y recreativa del municipio de tandil*. Buenos Aires, Argentina: Universidad del Salvador.
- Gunn, C. (1994). *Tourism planning*. London: Taylor & Francis.
- Heskett, J., Sasser, E., & Schlesinger, L. (1998). *The service profit chain*. Buenos Aires, Argentina: Gestion.
- Hilera, J. R., & Martínez, V. J. (1995). *Redes neuronales artificiales, fundamentos, modelos y aplicaciones*. Madrid, Spain: RA-MA Editorial.
- Ho, S.-J., & McKay, R. (2002). Balanced scorecard: Two perspectives. *The CPA Journal*, 72(3), 21-25.
- Indurkha, N., & Weiss, S. (1998). *Predictive data mining: A practical guide*. San Francisco: Morgan Kaufmann.
- Inmon, W. H. (1996). *Building the data warehouse*. New York: Wiley.
- Inmon, W. H., Glassey, K. L., & Welch, D. (1997). *Managing the data warehouse*. New York: Wiley.
- Inskip, E. (1991). *Tourism planning: An integral and sustainable development approach*. New York: Van Nostrand Reinhold.
- Kaplan, R., & Norton, D. (1999). *The balanced scorecard: Translating strategy into action*. Cambridge, MA: Harvard Business School Press.
- Kaplan, R., & Norton, P. (2001). Transforming the balanced scorecard from performance measurements to strategic management. *Accounting Horizons*, 15(1), 87.
- Kasabov, N. K. (1996). *Foundations of neural networks, fuzzy systems and knowledge engineering*. Cambridge, MA: MIT Press.
- Lazzari, L. L., Machado, E. A. M., & Pérez, R. H. (1998). *Teoría de la decisión fuzzy*. Buenos Aires, Argentina: Ediciones Macchi.
- Lopez, G. M. (1999). El cambio y la cultura organizacional en el diseño de un sistema de información de Gestión. *Gestión y Estrategia*, 15.
- Malina, M. A., & Selto, F. (2001). Communicating and controlling strategy: An empirical study of the effectiveness of the balanced scorecard. *Journal of Management Accounting Research*, 44, 47.
- Marakas, G. (1999). *Decision support systems*. Upper Saddle River, NJ: Prentice Hall.
- McIntyre, G. (1992). *Desarrollo turístico sostenible. Guía para los planificadores locales*. Madrid, Spain: Organización Mundial de Turismo.
- Miyake, D. (2002). Beyond the numbers: After years of evolution, balanced scorecard applications now integrate strategy and management for competitive advantage. *Intelligent Enterprise*, 15(12), 24-27.
- Neidorf, R. (2002, September/October). Knowledge management: Changing cultures changing attitudes. *Online*, 26(5), 60-63.
- Olvé, N., Roy, J., & Wetter, M. (1999). *Performance drivers: A practical guide to using the balanced scorecard*. New York: Wiley.
- Organización Mundial de Turismo. (1997). *Guía práctica para el desarrollo y uso de indicadores de turismo sostenible*. Madrid, Spain: Author.
- Probst, G., Raub, S., & Romhardt, K. (2000). *Managing knowledge*. Chichester, UK: Wiley.
- Simon, R. (1995, September/October). Los sistemas de control como instrumento de la renovación estratégica. *Harvard Deusto Business Review*.
- Tissen, R., Andriesen, D., & Lekanne Deprez, F. (2000). *The knowledge dividend*. New York: Prentice Hall.
- Van Lohuizen, C. W. W. (1996). Knowledge: Creation, diffusion, utilization. *Knowledge Management and Policy Making*, 8(1).
- Wisniewski, M., & Dickson, A. (2001). Measuring performance in Dumfries and Galloway constabulary with the balanced scorecard. *Journal of the Operation Research Society*, 52(10), 1057.

## **KEY TERMS**

**Balanced Scorecard Collaborative:** A strategic management system that measures, by means of quantitative relations of different selected variables, the behavior of the organization, taking into account the settled aims established in different perspectives (e.g., increase, internal processes, customers, finances). The analysis is

## **Knowledge Management in Tourism**

based on the cause–effect relations between the variables and ratios that represent them.

**Data Mining:** The process of the discovery of patterns, profiles, and significant trends through data analysis, making use of specific techniques such as neuronal networks and genetic and learning algorithms.

**Decision Support System:** Systems designed, built, and used to support the decision-making process. Its components are

- a. the data management system
- b. the model management system
- c. the knowledge engine
- d. the user interface
- e. the user or users

**Fuzzy Logic:** A procedure for analyzing approximate reasoning, which uses its imprecision and settles borderline cases with the concept of precision. Human reasoning is imprecise, and the ability to make reasonable decisions in such a clear environment of uncertainty is a major aspect that depends on the possibilities to obtain an approximate answer to some questions based on the acquired knowledge, which is normally inexact and not always reliable.

**Indicators:** Generally quantitative expressions that relate variables to defined criteria. They can be statistics, numbers, values, reasons, or other ways of representing information.

**Knowledge Management:** An organizational process that consists chiefly of the following stages:

- a. creating and generating knowledge
- b. organizing and attaching value to that knowledge
- c. transforming and transferring knowledge
- d. storing knowledge
- e. reusing knowledge

**Neuronal Networks:** Computing devices designed in such a way that they simulate nervous systems, with a great number of calculation elements that carry out no linear analogous functions. In computing, they can be used for forecasts, classification, detection of connections, or groupings.

**Online Analytical Processing (OLAP):** Systems of information for decision making in which the information is seen as cubes that consist of the combination of descriptive categories and qualitative values.

# Knowledge Mining

**Mahesh S. Raisinghani**

*Texas Woman's University, USA*

## INTRODUCTION

Numerous conferences and several articles in scholarly and business journals have tried to get a handle on knowledge. The growth of knowledge consulting organizations signals a growing conviction that knowing about knowledge is critical to business success. Multiple factors have led to the current knowledge boom. The perception and the reality of a new global competitiveness are one driving force. Rapid change and increasing competition for the dollars, marks, and yen of increasingly sophisticated consumers have led firms to seek a sustainable advantage that distinguishes them in their business environments.

Knowledge is neither data nor information, though it is related to both. Most people have an intuitive sense that knowledge is broader, deeper, and richer than data or information. More and more, business leaders and consultants talk about knowledge as the key to a sustainable competitive advantage. *Knowledge workers, knowledge-creating company, knowledge capital, and leveraging knowledge* have become familiar phrases (Davenport & Prusak, 1998; Turban, McLean, & Wetherbe, 2003).

During his keynote speech at the Information Resources' annual meeting in Boston, Massachusetts, Venkatraman (1998) discussed how companies manage their knowledge assets and how organizations have moved from the industrial economy to the knowledge economy (see Figure 1).

The purpose of this article is to discuss the knowledge concept of knowledge mining and address the following questions:

- How are the concepts of data life cycle and knowledge discovery related?

- What is the taxonomy of knowledge mining and its benefits?
- What is the role of knowledge in software development?

## BACKGROUND: DATA LIFE CYCLE AND KNOWLEDGE DISCOVERY

To better understand how to manage data and knowledge, it is necessary to trace how and where data flow in organizations. Businesses do not run on data, they run on information and their knowledge of how to put that information to use successfully. Everything from innovative product designs to brilliant competitive moves relies on knowledge. However, knowledge is not readily available. In many cases, it is continuously derived from data. However, because of the difficulties, a derivation may not be simple.

The transformation of data into knowledge may be accomplished in several ways. The process starts with data collection from various sources. These data are stored in a database followed by storage in a data warehouse. To discover knowledge, the processed data may go through a transformation that makes them ready for analysis. The analysis is done with data mining tools, which look for patterns, to support data interpretation. The result of all these activities is generated knowledge. Both the data, at various times during the process, and the knowledge, derived at the end of the process, may need to be presented to users by using different presentation tools. As illustrated in Figure 2, the created knowledge is stored in a knowledge base (Turban et al., 2003).

Figure 1. Transition from the industrial economy to the knowledge economy

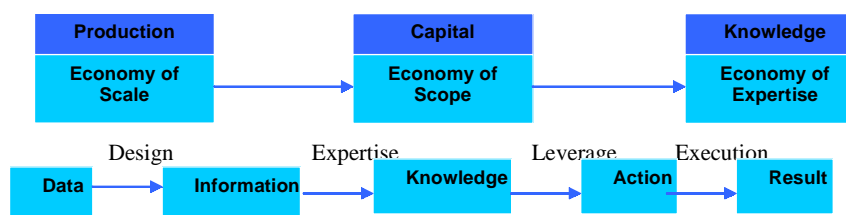
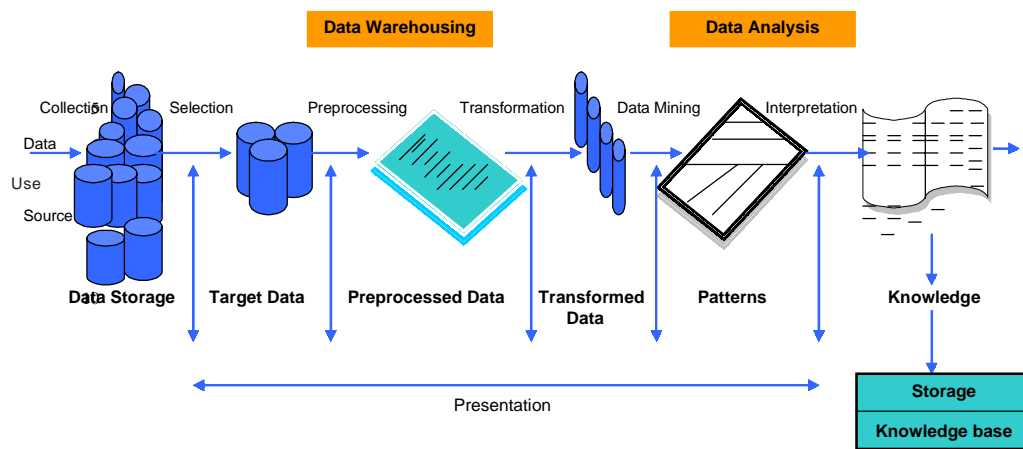


Figure 2. Data life cycle and knowledge discovery



## KNOWLEDGE MINING: TAXONOMY AND BENEFITS

All decision support systems use data, information, or knowledge. These three terms are sometimes used interchangeably. *Data* items refer to an elementary description about things, events, activities, and transactions that are recorded, classified, and stored but are not organized to convey any specific meaning. Data items can be numeric, alphanumeric, figures, sounds, or images. *Knowledge* consists of data items that are organized and processed to convey understanding, experience, accumulated learning, and expertise as they apply to current problem or activity (Grinstein, 2003; Grinstein, Kobsa, Plaisant, Shneiderman, & Stasko, 2003; Last, Friedman, & Kandel, 2003; Tan, Kumar, & Srivastava, 2004; Turban & Aronson, 1998).

The mental processing and representation of knowledge are complex activities, and our understanding is still rudimentary and subject to debate. A general concept for describing knowledge is an elusive as ever, though various key concepts have been developed from specific viewpoints in the cognitive sciences. Another way to define knowledge is to consider the way it is stored in human memory. Here, knowledge refers to a permanent structure of information stored in memory.

Han Koperski, Melli, Wang, and Zaane (1995) used the term *knowledge mining* as a practical synonym of *knowledge discovery*, not as an extension of it. At present, the use of the term is strongly associated as a synonym of *knowledge discovery and data mining*. In contrast, the term *software mining* is a special kind of

knowledge discovery wherein the source data is already in the form of rules or program code.

Knowledge mining consists of the following four integrated components, designed to seamlessly guide the extraction process and contribute to providing corporations with a concise understanding of their business rules (Aiken, Muntz, & Richards, 1994; Chiang, 1995; Weiss, Buckley, Kapoor, & Damgaard, 2003; Yang, Hongji, & Chu, 2001; Yang, Hongji, Chu, Cheng, & Zhan, 2001):

- *System-wide knowledge recovery* provides an overall view of the business processes supported by an application. The system-wide knowledge recovery facility enables analysts to identify the programs in which particular business rules exist and then extract those rules from applications.
- *Program-level analysis* enables the structure and interrelationships within programs to be revealed. The program focuses on variable usage, paragraph calls diagrams, GO TO diagrams, execution paths, and complex queries.
- *Business rule extraction* enables concise business rules to be extracted from within legacy programs and across entire legacy systems. Support is provided by
  - *variable-based techniques*, which enable a business rule to be extracted based upon a specific variable within a program.
  - *value-based specialization techniques*, which contain embedded data that can be greatly simplified and specific rules that can be uncovered.



- *system-wide techniques*, in which some legacy applications pass information from one program to another while processing a specific business rule. The system-wide extraction technique enables a business rule to be analyzed and extracted across multiple programs or across the system as a whole.
- *Automatic documentation* allows for a structured view of the application and generates books of HTML documentation. Knowledge mining's automatic documentation focuses on the ability to save diagrams and reports in various formats.

Knowledge mining satisfies the needs of corporations by helping to identify and understand the application assets that currently exist within their legacy applications. Knowledge mining:

- increases understanding of internal process. Knowledge mining helps corporations to unlock the business knowledge contained in their embedded rules, providing a better understanding of the internal processes that run their external business.
- increases reuse of existing assets. Increasingly, large portions of an organization's business rules can be reused in new platforms once identified and understood by the organization.
- decreases risk and cost of alternatives. Once a solid understanding of a corporation's business knowledge is obtained, organizations are then well positioned to provide input into the planning processes for legacy modernization initiatives, thus, decreasing the risk for determining incorrect alternatives for business needs, and increasing the value of their business processes.

For instance, RescueWare, a knowledge mining tool, enables organizations to identify and extract critical business rules embedded in legacy applications (Relativity Technologies, n.d.). The U.S. Air Force felt grounded because its core business processes (a key retail supply management system) was buried in a Unisys 2200 Clearpath mainframe computer, and its data was trapped in a proprietary DMS-100 database. The Air Force wanted to move to a flexible Web-based infrastructure to reduce information technology costs and to streamline its systems—all major priorities that could not be achieved with the inflexible structure of its legacy systems. Specifically, the Air Force wanted to Web-enable its standard base supply system (SBSS), a series of inventory, accounting, and order management systems that control the flow of supplies from the warehouse to deployment and integrate the SBSS system into the larger Air Force

Integrated Logistics System, a broad set of supply maintenance and accounting systems. It also wanted to provide direct support to all active Air Force units, the National Guard, and the Reserve.

RescueWare's automated functionality enabled the Air Force to reduce the expected project time to 30 months and reduce the cost to \$12 million—much lower than the investment that would have been required to complete this task manually. In the inventory and analysis phase of the Air Force project, RescueWare solutions were used to analyze and identify the key areas of value within the large, monolithic system containing approximately 1.6 million lines of code that had been built and refined by the Air Force over the course of three decades.

Then RescueWare's revolutionary knowledge mining capabilities isolated the Air Force's key business processes and extracted them from the larger application in which they had been contained. RescueWare's business rule extraction tools were used to create defined, stand-alone pieces called e-components that function as independent programs and can be easily integrated with other parts of the military's large technology framework. RescueWare was used to extract complexity information on these various components that proved important for managing the project and determining the best path in performing the tasks associated with the project (Relativity Technologies, n.d.). Knowledge mining would be able to analyze business rules and software code and extract more general rules or models from them. However, the machine learning techniques that would be necessary to deal with such complex input data, are still in its infancy.

## FUTURE TRENDS

Software development is knowledge intensive. Many concepts have been developed to ease or guide the processing of knowledge in software development, including information hiding, modularity, objects, functions and procedures, patterns and more. Methods and approaches in software engineering are often based on the results of empirical observations on individual success stories. The following table lists the viewpoint corresponding to each key knowledge concept in the cognitive sciences (Robillard, 1999).

Related studies have identified two types of knowledge-procedural and declarative—and their corresponding memory contents. Procedural knowledge, including psychomotor ability, is dynamic. Procedural memory stores all the information related to the skills developed to interact with our environment such as walking, typing, and so forth. Procedural knowledge



## Knowledge Mining

Table 1. Viewpoints corresponding to each key knowledge concept in the cognitive sciences

Key knowledge concept	Viewpoint
• Procedural / Declarative	Knowledge, nature of content
• Schema	Knowledge, integral structure of
• Proposition	Formal knowledge representation
• Chunking	Representing units of knowledge
• Planning	Managing knowledge structures

encompasses the know-how, and once learned is rarely forgotten.

Declarative knowledge, based on facts, is static and concerned with the properties of objects, persons, and events and their relationships. Declarative memory consists of two types of knowledge—topic or semantic, and episodic. Topic knowledge refers to the meaning of a word, such as its definitions in a dictionary or textbook. Episodic knowledge consists of one's experience with knowledge. These are learned through experience once the topic knowledge is obtained from textbooks, formal training, and education (Robillard, 1999).

Software development requires topic and episodic knowledge. The notion of “schema” was first proposed for artificial intelligence. The schema concept assumes that knowledge is stored in a human's memory in a preorganized way. A schema is a generic structure built up from an undefined variety of topics and from episodic knowledge. The topic part of the schema represents objects or events; the episodic part represents temporal or causal links between objects or events. For instance, our schema of the operating system represents our memory organization of the related items of topical knowledge, including icons, setup, layout, and menu structure. It also comprises episodic knowledge built up from user's experience with the operation system, including how to run a program, open a file, use a spreadsheet, listen to music, and send e-mail.

Knowledge formulation is based on atomic components described in terms of propositions and predicates. A proposition is the smallest unit of knowledge constituting an affirmation as well as the smallest unit that can be true or false. The theoretical hypothesis concerning the cognitive structure of human information system states that, at a certain level, information is organized in propositional form. Another component of the mental process is the amount of knowledge available for immediate processing. Psychologists use the concept of chunks ( $7 + \text{or} - 2$ ) to account for the limited amount of

knowledge that can be handled by the human mind at any given time. Software methodologies based on encapsulation, information hiding, modularization, abstraction, and even the divide-to-conquer approach all deal with the chunking phenomenon. Successful methodologies based on icons, graphic symbols, and reserved words are naturally limited to the chunk number for the simultaneous use of elements in working memory.

Planning is one of the human brain's most powerful natural activities. The limited capacity of the human mind's working memory cannot keep track of all the information from all the knowledge domains visited. These plans have three main characteristics (Robillard, 1999):

- heuristic nature
- optimal use of memory
- higher control level

Experience plays a major role in any knowledge activity. Psychologists recognize a distinct structure (i.e., episodic structure) in human memory that accounts for experience. Software development can be improved by recognizing the related knowledge structure or representation, including building schemas, validating schema default values, acquiring topic knowledge, performing planning activities, applying formal specifications to define problems, and having the appropriate tools to manage the chunking phenomenon.

## CONCLUSION

This article summarizes the elements necessary for the comprehension of knowledge mining. The transformation of data into knowledge to support decision making is a multiple step process and can be accomplished using different tools. Data mining and data warehousing play a major role in knowledge discovery. Software development is knowledge intensive and is based on the five key knowledge concepts in the cognitive sciences (i.e., procedural/declarative, schema, proposition, chunking, and planning). Mining knowledge at multiple concept levels may help end-users such as software analysts, managers/executives, or other decision-making personnel to find some interesting rules that are difficult to discover otherwise and view database contents at different abstraction levels and from different perspectives.

## REFERENCES

- Aiken, P., Muntz, A., & Richards, R. (1994). DoD legacy systems: Reverse engineering data requirements. *Communications of the ACM*, 37(5), 26-41.
- Chiang, R. H. L. (1995). A knowledge-based system for performing reverse engineering of relational database. *Decision Support Systems*, 13, 295-312.
- Davenport, T. H., & Prusak, L. (1998). *Working knowledge*. Cambridge, MA: Harvard Business School Press.
- Grinstein, G. (2003, October). Integrating visualization with data mining and knowledge discovery for high dimensional data exploration and discovery. *Proceedings of the IEEE Visualization Conference*, Seattle, WA.
- Grinstein, G., Kobsa, A., Plaisant, C., Shneiderman, B., & Stasko, J. (2003, October). Which comes first, usability or utility? *IEEE Visualization Conference Proceedings*, Seattle, WA.
- Han, J., Fu, Y., Koperski, K., Melli, G., Wang, W., & Zaane, O. (1995). Knowledge mining in databases: An integration of machine learning methodologies with database technologies. Available online from <http://citeseer.ist.psu.edu/han95knowledge.html>
- Last, M., Friedman, M., & Kandel, A. (2003, August 24-27). The data mining approach to automated software testing. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC (pp. 388-396).
- Relativity Technologies. (n.d.) USAF saves millions with supply management software. Retrieved June 29, 2004, from <http://64.233.161.104/search?q=cache:xdJpf6az-ZEJ:www.relativity.com/News/coverage/eCompany-May-01.htm+RescueWare+software&hl=en>
- Robillard, P. N. (1999). The role of knowledge in software development. *Communications of the ACM*, 42(1), 87-92.
- Tan, P.-N., Kumar, V., Srivastava, J. (2004). Selecting the Right Objective Measure for Association Analysis Information Systems. *Information Systems Archive*, 29(4), 293-313.
- Turban, E., McLean, J., & Wetherbe, J. (2003). *Information technology for management, making connections for strategic advantage* (3rd Ed.). New York: Wiley.
- Turban, E., & Aronson, R. (1998). *Decision support systems and intelligent systems* (5th Ed.). Upper Saddle River, NJ: Prentice Hall.
- Venkatraman, N. (1998). *Keynote speech*. Annual Meeting of the International Resources Management Association, Boston.
- Weiss, S. M., Buckley, S. J., Kapoor, S., & Damgaard, S. (2003). Knowledge-based data mining. *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC, August 24-27 (pp. 456-461).
- Yang L., Hongji, Y. & Chu. W. (2001, January/February). A concept-oriented belief revision approach to domain knowledge recovery from source code. *Journal of Software Maintenance and Evolution: Research and Practice*, 13(1), 31-52.
- Yang L., Hongji, Y., Chu, W., Xiaochun, C. & Zhan, C. (2001). Improving the reliability of knowledge mining in legacy code by utilising cooperative information [Special issue]. *International Journal of Fuzzy Systems*, 3(2).

## KEY TERMS

**Automatic Documentation:** Allows for a structured view of the application and generates books of HTML documentation. Knowledge mining's automatic documentation focuses on the ability to save diagrams and reports in various formats.

**Business Rule Extraction:** Enables concise business rules to be extracted from within legacy programs and across entire legacy systems.

**Declarative Knowledge:** Based on facts, is static and concerned with the properties of objects, persons, and events and their relationships.

**Episodic Knowledge:** Declarative memory consists of two types of knowledge—topic or semantic, and episodic. Episodic knowledge consists of ones experience with knowledge. These are learned through experience once the topic knowledge is obtained from textbooks, formal training, and education.

**Knowledge Mining:** A practical synonym of *knowledge discovery*, not an extension of it. At present, the use of the term is strongly associated as a synonym of *knowledge discovery and data mining*. Knowledge mining consists of the following four integrated components designed to seamlessly guide the extraction process and contribute to providing corporations with a concise understanding of their business rules: System-wide knowledge recovery, program-level analysis, business rule extraction, and automatic documentation.

## **Knowledge Mining**

**Program-Level Analysis:** Enables the structure and interrelationships within programs to be revealed. The program focuses on variable usage, paragraph calls diagrams, GO TO diagram, execution paths, and complex queries.

**Software Mining:** A special kind of knowledge discovery in which the source data is already in the form of rules or program code.

**System-Wide Knowledge Recovery:** Provides an overall view of the business processes supported by an application. The system-wide knowledge recovery facility enables analysts to identify the programs in which particular business rules exist and then extract those rules out from across applications.

**Topic Knowledge:** Declarative memory consists of two types of knowledge-topic or semantic and episodic. Topic knowledge refers to the meaning of a word, such as its definitions in a dictionary or textbook.

**K**

# Logic Databases and Inconsistency Handling

**José A. Alonso-Jiménez**

*Universidad de Sevilla, Spain*

**Joaquín Borrego-Díaz**

*Universidad de Sevilla, Spain*

**Antonia M. Chávez-González**

*Universidad de Sevilla, Spain*

## INTRODUCTION

Nowadays, data management on the World Wide Web needs to consider very large knowledge databases (KDB). The larger is a KDB, the smaller the possibility of being consistent. Consistency in checking algorithms and systems fails to analyse very large KDBs, and so many have to work every day with inconsistent information.

Database revision—transformation of the KDB into another, consistent database—is a solution to this inconsistency, but the task is computationally untractable. Paraconsistent logics are also a useful option to work with inconsistent databases. These logics work on inconsistent KDBs but prohibit undesired inferences. From a philosophical (logical) point of view, the paraconsistent reasoning is a need that the self human discourse practises. From a computational, logical point of view, we need to design logical formalisms that allow us to extract useful information from an inconsistent database, taking into account diverse aspects of the semantics that are “attached” to deductive databases reasoning (see Table 1). The arrival of the semantic web (SW) will force the database users to work with a KDB that is expressed by logic formulas with higher syntactic complexity than are classic logic databases.

## BACKGROUND

Logic databases are based on the formalisms of first order logic (FOL); thus, they inherit a classical semantics that is based on models. Also, they can be interpreted within a proof-theoretic approach to logical consequence from the logic programming paradigm (Lloyd, 1987). The extended database semantics paradigm is developed to lay before the foundations of query-answering tasks and related questions (see Minker, 1999), but its aim is not to deal with inconsistencies. The data cleaning task may involve—in the framework of repairing logic databases—

*Table 1. Semantics aspects to consider in logic databases*

- |  |
|--|
| <ul style="list-style-type: none"> <li>• Classical semantics for FOL</li> <li>• Extended semantics for databases</li> <li>• Reiter's formalization of databases (Reiter, 1984). Closed world assumption</li> <li>• Relations among a KDB, queries and integrity constraints</li> <li>• Expressive power of recursive definitions</li> <li>• Consistency checking versus intentional part of the KDB</li> <li>• Multivalued semantics</li> <li>• Contextualized semantics for ontologies or data</li> </ul> |
|--|

logical reasoning and automated theorem proving (Boskovitz, Goré, & Hegland, 2003).

On the other hand, new paradigms, such as SW, need new formalisms to reason about data. Description logics (DL) provide logic systems based on objects, concepts, and relationships, with which we can construct new concepts and relations for reasoning (Baader, Calvanese, McGuinness, Nardi, & Patel-Schneider, 2003). Formally, DL are a subset of FOL, and the classical problems on consistency remains, but several sublogics of DL provide nice algorithms for reasoning services. The ontology Web language (OWL; its DL-sublanguage) is a description logic designed for automated reasoning, not only designed for the classical ask–tell paradigm. With languages such as OWL, ontologies exceed their traditional aspects (e.g., taxonomies and dictionaries) to be essential in frameworks as data integration.

The classical notion of inconsistency in databases mainly deals with the violation of integrity constraints. This notion must be expanded because of the new notion of logic databases in SW, in which ontologies and data both play the same role in knowledge management. Therefore, there are several sources of inconsistency (see Table 2). This role is not only limited to the database but also includes the verification and validation task of knowledge-based systems (Bench-Capon, 2001). Inconsistency arises in the initial steps of ontology building due to several reasons and not only by the updating of data. In general, the repair of a logic database involves the study

Table 2. A list of sources of inconsistencies from the practical knowledge management

<p><b>Dirty data</b> Some kinds of data dirtiness give rise to fail of integrity constraints (Kim, Choi, Hong, Kim, &amp; Lee, 2003).</p> <p><b>Neglected development of the intentional database</b> The development of the intentional component part of the database produces an inconsistent theory (the <i>ontology</i>, in the SW paradigm. See, e.g., Backlawski, Kokar, Waldinger, &amp; Kogut, 2002).</p> <p><b>Logical interpretation in data integration</b> The design of a data integration system—to provide uniform access to multiple and heterogeneous information sources—needs of query reformulation, ontology mapping, or integration and, in general, logical interpretation.</p> <p><b>Procedural incompleteness</b> Incomplete query-answering algorithms do not produce any witness for some integrity constraint of existential character.</p> <p><b>Conflict information in data integration</b> Special case in data integration: The information that is received from different consistent resources is inconsistent.</p> <p><b>Deficient specification of the ontology language</b> The specification of the language for ontology representation is inconsistent (Fikes, McGuinness, &amp; Waldinger, 2002).</p> <p><b>Inadequate data cleaning</b> Some criteria to take decisions in data cleaning make the KDB inconsistent.</p>	<p><b>Deficient maintenance of KDB</b> The kind of the selected method for preserving consistency is not robust under every sort of updates.</p> <p><b>Wrong data mining</b> The output of data mining systems does not satisfy integrity constraints or ontology requirements. In multiagent data mining, the different outputs lead to a problem of data integration.</p> <p><b>Expressiveness clashes with model theory</b> The logical syntax/semantics (from deductive database paradigm) does not allow new features to be used in the usual knowledge representation in the SW. This absence implies messy definitions that may be incorrect.</p> <p><b>Bad design of the common knowledge shared by different users</b> The intentional component part does not describe the users intended requirements. The logical consistent KDB does not fit with users's beliefs. Thus, new updates may produce inconsistencies.</p> <p><b>Deficient ontology learning</b> The ontology acquisition is a tedious task that the user tends to finish before he or she thinks as advisable. A poor ontology associated to consistent data may produce inconsistency.</p> <p><b>Skolem Noise</b> A kind of dirty data produced by the use of an automated theorem prover in data cleaning of logic databases (Alonso-Jiménez, Borrego-Díaz, Chávez-González, Gutiérrez-Naranjo, &amp; Navarro, 2003).</p>
---	--

of the soundness and perhaps completeness (i.e., the method output's only correct solutions and all the relevant solutions). Semantics would support reasoning services such as self-consistency, checking the relations between concepts (as subsumption), and classification of objects according to the ontology.

Systems exist in which both paradigms, classical and SW logic databases, are conciliated under the extension of the former (see, e.g., Pan & Heflin, 2003). However, the relation between DL and database models may not be fruitfully formalized because of the limited expressiveness of the DL system selected to make the reasoning feasible (see chapter 4 in Baader et al., 2003).

## INCONSISTENCY HANDLING

Solutions that are suggested to work in the presence of inconsistencies can be classified according to different views (see Table 3, where several references appear). The first aspect—and maybe the most important—is the compatibility between the original semantics of the KDB source and the logical formalism selected to handle inconsistency. From this point of view there exist paraconsistent logics that limit the inference power of FOL to avoid undesired answers and also modal logics for representing different aspects of the information sources. These

approaches manage semantics that are essentially different from the semantics of KDBs. On the other side we can find methods that classify, order, or both, interesting subsets according to the original semantics of the KDB, such as the argumentative approach or the integration of data by fusion rules, but they do not repair the KDB. Other methods also exist that propose how the KDB should be revised (e.g., integrity constraints of the extensional database). However, it is necessary to point out that the automated knowledge revision is an essentially different task in the case of ontologies, because the ontology source represents a key organisation of the knowledge of the owner and, as in every logical theory, minor changes may produce unexpected and dangerous anomalies.

Another point of view concerns the share of the KDB that is repaired when an anomaly is found. According to this, the methods based on arguments mentioned earlier can be used to repair only the anomalous argument. Due to the high complexity of consistency checking algorithms, to preserve consistency under updates is a better option than repairing. In the case of evolving ontologies, new systems such as KAON infrastructure are needed (for more information, see <http://kaon.semanticweb.org/kaon>).

There are methods dealing with the enforcement of consistent answers (i.e., answers that satisfies integrity constraints) from inconsistent databases; it is done by



Table 3. List of recent solutions for inconsistency handling

- Paraconsistent logics (Grant & Subrahmanian, 2000; Hunter, 1998)
- Nonrepairing and merging-oriented techniques:
  - Preorders on information sets (Cantwell, 1998; Marquis & Porquet, 2003 in the paraconsistent framework)
  - Argumentative hierachy (Elvang-Goransson & Hunter, 1995), argumentative frameworks (Dung, 1995) and databases (Pradhan, 2003)
  - Fusion rules (Bloch et al., 2001)
  - Merging databases (Cholvy & Moral, 2001)
  - Contextualizing ontologies (Bouquet, Giunchiglia, van Harmelen, Serafini, & Stuckenschmidt, 2003) and data (MacGregor & Ko, 2003)
- Measuring the anomalies:
  - Evaluating by means of a paraconsistent logic (Hunter, 2003)
  - Measuring inconsistent information (Knight, 2003)
  - Consistent interpretation of Skolem noise (Alonso et al., 2003)
- Repairing techniques:
  - To apply knowledge reductions in inconsistent systems (Kryszkiewicz, 2001)
  - Fellegi-Holt method (Boskovitz et al., 2003)
  - Database repairs by tableaux method (Bertossi & Schwind, 2004)
  - Consistent querying to repair databases (Greco & Zumpano, 2000)
- Consistent enforcement of the database by means of greatest consistent specializations (Link, 2003)
  - Consistent answering techniques without reparation:
    - Transformation of the query to obtain consistent answers (Celle & Bertossi, 1994)
    - Consistent query answer in the presence of inconsistent databases (Greco & Zumpano, 2000)
    - To use bounded paraconsistent inference (see, e.g., Marquis & Porquet, 2003)
    - Detecting the cause of the inconsistency and retrieving a subset of the original KB (Ariel & Avron, 1999)
- Consistency preserving methods:
  - Consistency preserving updates in deductive databases (Mayol & Teniente, 2003)

transforming the self-query or by limiting the inference power of the system. Another method is to work in the context of data integration–merging (Levy, 2000). Fusion rules are the most direct treatment of simple information sets. The complex case, in which several ontologies comes into play, can be solved by contextualizing the knowledge. The contextualization of ontologies is an extension of the classical method introduced by McCarthy, and it has been used in important ontology projects such as CyC (for more information, see <http://www.cyc.com>). The use of contexts prevents inconsistencies and it allows to build coherent subsets of the ontology target.

Finally, there exist measures to *estimate inconsistency*. Although these measures may be unfeasible by their semantic oriented definitions, this obstacle may be partially solved by weakening metrics that estimate the cognitive difference between the ontology source and the ontology target by using only syntactic features (see, e.g., Gutiérrez-Naranjo, Alonso-Jiménez, & Borrego-Díaz, 2002; Hunter, 2003).

## FUTURE TRENDS

To handle inconsistencies in the semantic web, future trends must study verification techniques based on sound, limited testing and aided by a powerful automated theorem prover (see Alonso-Jiménez et al., 2003; Boskovitz et al., 2003). These techniques need a deep analysis of the behaviour of automated theorem provers having a great autonomy, because a slanted behaviour may produce defficient reports about inconsistencies in the KDB.

## CONCLUSION

Inconsistency handling has been a prevailing task in important fields as the semantic web, data integration, and data cleaning. Several techniques are proposed, but the need of working with very large databases makes some of them unfeasible, especially those that are applied on the full KDB.

## REFERENCES

- Alonso-Jiménez, J. A., Borrego-Díaz, J., Chávez-González A., Gutiérrez-Naranjo M. A., & Navarro-Marín, J. D. (2003). Towards a practical argumentative reasoning in qualitative spatial databases. *Proceedings of the 16th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE 2003)*, *Lecture Notes in Computer Science*, 2850 (pp. 789-798).
- Ariel, O., & Avron, A. (1999). A model-theoretic approach for recovering consistent data from inconsistent knowledge bases. *Journal of Automated Reasoning*, 22(2), 263-309.
- Baader, F., Calvanese, D., McGuinness D., Nardi, D., & Patel-Schneider, P. (2003). *The description logic handbook*. Cambridge, UK: Cambridge University Press.
- Baclawski, K., Kokar, M., Waldinger, R., & Kogut, P. (2002). Consistency checking of semantic web ontologies. *Proceedings of the First International Semantic Web Conference 2002 (ISWC'02)*, *Lecture Notes in Computer Science*, 2342 (pp. 454-459).
- Bench-Capon, T. (2001). The role of ontologies in the verification & validation of knowledge-based systems. *International Journal of Artificial Intelligence*, 16, 377-390.
- Bertossi, L. E., & Schwind, C. (2004). Database repairs and analytic tableaux. *Annals of Mathematics and Artificial Intelligence*, 40(1/2), 5-35.
- Bloch, I., Hunter, A., Appriou, A., Ayoun, A., Benferhat, S., Besnard, P., Cholvy, L., Cooke, R., Cuppens, F., Dubois, D., Fargier, H., Grabisch, M., Kruse, R., Lang, J., Moral, S., Prade, H., Saffiotti, A., Smets, P., Sossai, C. (2001). Fusion: General concepts and characteristics, *International Journal of Intelligent Systems*, 16(10), 1107-1134.
- Boskovitz, A., Goré, R., & Hegland, M. (2003). A logical formalisation of the Fellegi-Holt method of data cleaning. *International Conference on Intelligent Data Analysis (IDA 2003)*, *Lecture Notes in Computer Science*, 2810 (pp. 554-565).
- Bouquet, P., Giunchiglia F, van Harmelen F., Serafini, L., & Stuckenschmidt, H. (2003). C-OWL: Contextualizing ontologies. *Proceedings of the 2nd International Semantic Web Conference 2003 (ISWC'03)*, *Lecture Notes in Computer Science*, 2870 (pp. 164-179).
- Cantwell, J. (1998). Resolving conflicting information. *Journal of Logic, Language and Information*, 7(2), 191-220.
- Cholvy, L., & Moral, S. (2001). Mergin databases: Problems and examples. *International Journal of Intelligent Systems, Special Issue on Data and Knowledge Fusion*, 16(10).
- Celle, A., & Bertossi, L. (1994). Consistent data retrieval. *Information Systems*, 19(4), 33-54.
- Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and  $n$ -person games. *Artificial Intelligence*, 77(2), 321-358.
- Elvang-Goransson, M., & Hunter, A. (1995). Argumentative logics: Reasoning from classically inconsistent information. *Data and Knowledge Engineering*, 16(1), 125-145.
- Fikes, R., McGuinness, D. L., & Waldinger, R. (2002, January). A first-order logic semantics for semantic Web markup languages [Knowledge Systems Laboratory Tech. Rep. No. 02-01]. Retrieved September 16, 2004, from [http://www.ksl.stanford.edu/KSL\\_Abstracts/KSL-02-01.html](http://www.ksl.stanford.edu/KSL_Abstracts/KSL-02-01.html)
- Grant, J., & Subrahmanian, V. S. (2000). Applications of paraconsistency in data and knowledge bases. *Synthese*, 125, 121-132.
- Greco S., & Zumpano, E. (2000) Querying inconsistent databases. *Proceedings of the 7th International Conference of Logic for Programming and Automated Reasoning (LPAR 2000)*, *Lecture Notes in Computer Science*, 1955 (pp.308-325).
- Gutiérrez-Naranjo, M. A., Alonso-Jiménez, J. A., & Borrego-Díaz, J. (2003). A quasimetric for machine learning. In F. J. Garijo, J. C. Riquelme, & M. Toro (Eds.), *Advances in Artificial Intelligence (IBERAMIA 2002)*, *Lecture Notes in Computer Science*, 2527 (pp. 193-203).
- Hunter, A. (1998). Paraconsistent logics. In D. Gabbay & Ph. Smets (Eds.), *Handbook of defeasible reasoning and uncertain information* (pp. 13-44). Dordrecht: Kluwer.
- Hunter, A. (2003). Evaluating the significance of inconsistencies. *Proceedings of the International Joint Conference on AI (IJCAI'03)* (pp.468-473). San Francisco: Morgan Kaufmann.
- Kim, W. Y., Choi, B.-J., Hong, E. K., Kim, S.-K., & Lee, D. (2003). A taxonomy of dirty data. *Data Mining Knowledge Discovery*, 7(1), 81-99.
- Knight, K. M. (2003). Two information measures for inconsistent sets. *Journal of Logic, Language and Information*, 12(2), 227-248.

Kryszkiewicz, M. (2001). Comparative study of alternative types of knowledge reduction in inconsistent systems. *International Journal of Intelligent Systems*, 16(1), 105-120.

Levy, A. (2000). Logic-based techniques in data integration. In J. Minker (Ed.), *Logic-based artificial intelligence* (pp. 575-596). Dordrecht: Kluwer.

Link, S. (2003). Consistency enforcement in databases. *Proceedings of the 2nd International Workshop on Semantics in Databases 2001, Lecture Notes in Computer Science*, 2582 (pp.139-159).

Lloyd, F. (1987). *Foundations of logic programming*. Berlin: Springer.

MacGregor, R., & Ko, I.-Y. (2003). Representing contextualized data using semantic Web tools. *Electronic Proceedings of the ISWC'03 Workshop on Practical and Scalable Semantic Web Systems Services*. Retrieved September 16, 2004, from <http://km.aifb.uni-karlsruhe.de/ws/psss03/proceedings/macgregor-et-al.pdf>

Marquis, P., & Porquet, N. (2003). Resource-bounded paraconsistent inference. *Annals of Mathematics and Artificial Intelligence*, 39(4), 349-384.

Mayol, E., & Teniente, E. (2003). Consistency preserving updates in deductive databases. *Data and Knowledge Engineering*, 47(1), 61-103.

Minker, J. (1999). Logic and databases: A 20 year retrospective. In F. Pirri & H. Levesque (Eds.), *Logical foundations for cognitive agents* (pp. 234-299). Berlin: Springer.

Pan, Z., & Heflin, J. (2003). DLDB: Extending relational databases to support semantic web queries. *Electronic Proceedings of the ISWC'03 Workshop on Practical and Scalable Semantic web Systems Services*. Retrieved September 16, 2004, from <http://km.aifb.uni-karlsruhe.de/ws/psss03/proceedings/pan-et-al.pdf>, <http://km.aifb.uni-karlsruhe.de/ws/psss03/proceedings/pan-et-al.pdf>, and <http://km.aifb.uni-karlsruhe.de/ws/psss03/proceedings/pan-et-al.pdf>

Pradhan, S. (2003). Connecting databases with argumentation. *Web Knowledge Management and Decision Support, 14th International Conference on Applications of Prolog, (INAP 2001), Lecture Notes in Computer Science*, 2543 (pp.170-185).

Reiter, R. (1984). A logical reconstruction of relational database theory. In J. L. Mulopoulos & J. W. Schmidt (Eds.), *On conceptual modelling* (pp. 163-189). New York: Springer.

## KEY TERMS

**Arguments:** An argument in a KDB is a pair  $(\Delta, F)$ , where  $\Delta$  is a subset of the KDB such that  $\Delta$  entails  $F$ . The basic relation among arguments is *rebutting*.

**Closed World Assumption:** A principle that claims that every atom not entailed by the KDB is assumed to be false. This principle is sound on KDBs with simple syntax, as logic programs.

**Contextualizing Logics:** Method to formally represent knowledge associated with a particular circumstance on which it has the intended meaning.

**Description Logics:** Logical formalism to represent structured concepts and the relationships among them. Formally, it is a subset of FOL dealing with *concepts* (monadic predicates) and *roles* (binary predicates) which are useful to relate concepts. KDB in DL are composed of a *Tbox* (the intensional component) and an *Abox* (box of asserts, the extensional component part).

**Logical Inconsistency:** A logical theory is inconsistent if there is no logical models for it. In the logic database paradigm, the notion of inconsistency is usually restricted to express the violation of integrity constraints. This restriction makes sense when the data are atomic formulas.

**Ontology Web Language (OWL):** Language (based on description logics) designed to represent ontologies capable of being processed by machines. The World Wide Web Consortium released OWL as recommendation (see <http://www.w3.org/2001/sw/webOnt>).

**Paraconsistent Logics:** Logic systems that limit, for example, the power of classical inference relations to retrieve non trivial information of inconsistent sets of formulas.

**Reiter's Formalization of Database Theory:** A set of axioms that, when added to a relational database, formalize the reasoning with them. They are the *Unique Names Principle*, the *Domain Closure Axiom*, the *Completion Axioms* and *Equality Axioms*. This formalization permits identification of the answering with logical consequences.

**Skolem Noise:** A kind of anomalous answers obtained by resolution oriented theorem provers when it works on nonclausal theories. The classical method of skolemization leads to new function symbols with no intended meaning. If these symbols appear in the output, the answer may not be consistently interpreted.

# Main Memory Databases

**Matthias Meixner**

*Technische Universität Darmstadt, Germany*

## INTRODUCTION

For a long time, hard disks were the only technology that could store enough information to hold a database and offered random access at the same time. Therefore, conventional database management systems were tuned to take this technology to the maximum. But in recent years, main memories have become cheaper and grown to a point that for some fields of application it allows one to keep the whole information of a database in main memory and therefore speed up operation. This article focuses on the differences in conventional databases that affect both performance and internal structure of a database management system.

## BACKGROUND

When storing information in main memory, many design decisions that were based on disk storage are not valid anymore, and different steps have to be taken to achieve maximum performance since main memory and disk storage have very different access performance characteristics. The access time of main memory is of orders of magnitude smaller than for disk storage, but on the other hand main memory is volatile, while disk storage is not. Disk accesses exhibit a high fixed cost per access due to seek time. Therefore, to achieve good performance, accesses should be sequential and transfer large amounts of needed data, i.e., data layout on disk is critical. In contrast, the access time of main memory is less dependent on the location, and therefore data layout is far less critical, although this is also changing since the access time of on-processor caches is improving faster than the access time of main memory. These differences have effects on almost every aspect of database management. We will discuss this in the following sections.

## MAIN THRUST

### Main Differences to Disk Resident Databases

Both conventional disk resident database management systems (DRDBMSs) and main memory database man-

agement systems (MMDBMSs) process data in main memory, and both keep a (backup) copy on disk. If the cache of a DRDBMS is large enough, sooner or later the whole database will be in cache, and an MMDBMS needs to store a backup copy on disk so that it is able to recover from failures. So what is the main difference between a DRDBMS and an MMDBMS? The key difference is that in an MMDBMS the primary copy of the database lives permanently in main memory (Garcia-Molina & Salem, 1992), and this has important implications for the algorithms and data structures used. Even if the whole database of a DRDBMS is cached in main memory, it will not provide best performance since a DRDBMS is not tuned for this case. A DRDBMS cannot rely on data being present in main memory at all times. Therefore, each data access has to go to the buffer manager to make sure data is still in main memory. Index structures in DRDBMSs are designed for disk access and may trade computing power and storage efficiency for a lower number of disk accesses since disk access is the processing-time dominating factor in DRDBMSs. This overhead incurs even if all data is cached in main memory. In MMDBMSs the situation is different: Since data is guaranteed to stay present in main memory, index structures and all the other parts of the database do not need to consider disk access and can be tuned for low computational cost.

### Performance and Data Structures

In DRDBMSs disk accesses have traditionally been the bottleneck since in the time required for one single disk access, a processor can perform up to several million instructions. Therefore, the most important optimizations in disk-based systems are to reduce the number of disk accesses, to prefer sequential access, and to keep the processor busy while waiting for I/O. To reduce the number of disk accesses, caching is used, and special index structures, like the B+ tree, were developed. Data that is used together in a database is grouped on disk to be able to access it using one single sequential read. A high degree of concurrency is employed to keep the processor busy while other transactions are waiting for I/O, and therefore a small locking granularity down to record level locking is used.

In the case of MMDBMSs all these optimizations are not relevant anymore; instead, performance is only determined by the CPU efficiency of the algorithms used.



MMDBMSs have an edge over disk-based systems in respect thereof since many simplifications can be used that reduce the processing power required when data is guaranteed to be in main memory. A DRDBMS cannot rely on data being cached in main memory. Therefore, each access has to go to the buffer manager to bring data into main memory or to make sure that data is already in main memory. In an MMDBMS this step does not exist and therefore does not consume processing power. Pointers can be used to address data: Instead of storing attribute values in an index, pointers to it can be used, reducing the size of the index and reducing the complexity of dealing with long or variable length fields.

Traditional index structures are also affected: Since disk accesses do not occur any more, index structures can be tuned for a low consumption of main memory and processing power. One result of this is the T-tree (Lehman & Carey, 1986). It is a binary search tree whose nodes contain more than one item to reduce the number of calls to the memory management and to reduce the amount of memory consumed by pointers.

Since a new access time gap between on-processor cache and main memory opens up, data layout becomes more critical again. Cache-sensitive search trees (CSS-trees; Rao & Ross, 1999) address this problem and optimize the cache performance to achieve an additional speedup.

Pointers can also be passed to the applications, eliminating temporary copies of data. While this may boost performance, there is the danger that an application modifies unauthorized parts of the database. This can be circumvented by using a special compiler for the transactions that enforces checking of proper authorization and logs every object modification (Garcia-Molina & Salem, 1987).

## Commit Processing

From a user's point of view MMDBMSs should not differ from conventional database systems apart from the higher performance. Therefore MMDBMSs must be able to support the ACID properties known from conventional database systems. This has some important implications that will be described in the following sections.

## Recovery

A commit must be able to guarantee persistence. Since data in main memory is volatile, some kind of mechanism must exist that protects data in case of a failure. One common mechanism is to use logging to disk alongside using backup copies of the data. In this case before a transaction can commit, its activity records must be writ-

ten to the log. This can greatly affect response time since each transaction must wait for at least one stable write before committing. Although this is also the case in DRDBMSs, it is more severe in main memory systems because the logging represents the only disk access. Therefore, logging may become the bottleneck for the whole system. This problem can be solved by using precommitting and group-commit (DeWitt et al., 1984). Precommitting releases locks as soon as the log records are placed in the log but before they have actually been written to disk. By this the blocking delay of a transaction is reduced, but the response time of this transaction is not improved. Group-commit accumulates log records of several transactions and flushes them to disk in one write operation. While the number of I/O operations is reduced to relieve the I/O bottleneck, the response time may be even increased due to the accumulation of log records.

Since disk access is the bottleneck, the key to a faster response time is to eliminate the disk access and yet be able to guarantee persistence. One very expensive solution is to use solid-state disks that represent a drop-in replacement for conventional hard disks but that store data in battery (and disk) backed-up DRAM and therefore are able to offer very fast access time. Flash ROM cannot be used as a logging device due to the limited number of writes supported by this memory technology. The write-intensive use would wear out Flash-ROM within weeks and months depending on the actual use (assuming 1,000,000 supported write cycles—current numbers are closer to 100,000—and one write per 10 seconds to the same place in memory, flash ROM would last for about 120 days).

Another solution to guarantee persistence is by logging over a network using, e.g., a distributed, fault-tolerant write cache (Mao, Zhang, Wang, & Zheng, 2002). Probably the most extreme solution is to guarantee persistence by means of fault tolerance in a distributed system. In case of a failure, data is not lost, but at least one copy still exists in main memory of one node of the distributed system. This not only allows improved response time, since network access is much faster than disk access, but at the same time such a system is able to provide high availability (Meixner & Buchmann, 2004).

## Backup

Main memory is volatile and is lost in case of a power failure. Unless special care is taken, like in Bressoud, Clark, and Kan (2001), data is not only lost after a failure but even during a simple restart of the system. Therefore, a backup copy of the database must exist. One obvious solution is to back up data to disk(s). In case of a failure this backup can be used together with the logs to recover



the database. To keep the time needed for a recovery short, the backup must be kept up-to-date. One commonly used mechanism is checkpointing (Woo, Kim, & Lee, 1997). Since the backup disk is only accessed by the checkpoint that backs up data in the background and no application has to wait for I/O to this disk, the I/O can be tailored to meet the need of the checkpoint alone. Therefore, a very large block size can be used since large blocks can be written to disk more efficiently.

Backing up MMDBMSs has some problems not found in DRDBMSs: While in DRDBMSs data already has a well-known location and structure on disk, this is not the case in MMDBMSs. In contrast an MMDBMS has to answer the questions: How and where should data be stored, and how should pointers be treated? Solutions to this problem are presented in Lin and Dunham (1996) and Salem and Garcia-Molina (1989).

### Concurrency Control

As it was mentioned above, in DRDBMSs a high degree of concurrency is desirable to keep the processor(s) busy while other transactions are waiting for disk I/O. To achieve maximum parallelism, locking granularity should be as small as possible. In an MMDBMS waiting for disk I/O is not an issue. Therefore, the higher overhead of locking outweighs the advantage of small locking granules. It has been suggested that very large lock granules, e.g., relations or even the entire database, are most appropriate. Even a serial execution of transactions that eliminates the need for locking can be advantageous in MMDBMSs (Blott & Korth, 2002).

### Related Issues: Real-Time Systems

The goal in real-time systems is to guarantee the completion of a task before a deadline. Therefore, databases in real-time systems must estimate the worst-case runtime of all transactions to be able to schedule them so that every task meets its deadline. DRDBMSs have a problem regarding the worst-case runtime of transactions: In the worst case, each access may result in disk I/O, resulting in a very high worst-case runtime; however, in average, the processing time is much shorter. In MMDBMSs the situation is different: At most, one single write access to disk is required, i.e., writing the log record. Therefore, the worst-case processing time is much better. To resolve this problem in DRDBMSs prefetching can be used, which preanalyzes the transactions to be run and constructs a dynamic transaction-oriented main memory sub-database by preloading all potentially required data into main memory to profit from the advantages of MMDBMSs (Wedekind & Zoerntlein, 1986).

Another advantage of MMDBMSs results from the coarser locking granularity. If tuples are locked on demand, deadlocks may occur. Since the tuples to be locked are not known in advance, deadlocks cannot be avoided. This is not acceptable in real-time systems since deadlocks screw up any runtime estimation. On the other hand if whole relations are locked, the relations to be locked may be determined in advance since the relations involved are listed in the transaction. This allows running only those transactions in parallel that do not have conflicts, thus avoiding deadlocks completely.

### FUTURE TRENDS

For some fields of applications, MMDBMSs offer significant advantages over DRDBMSs systems. But some of the mechanisms developed for MMDBMSs can also be integrated in DRDBMSs: For example the advantage of pointers can be exploited by using the page fault mechanism, to swizzle (convert) the pointers to an in-memory representation as soon as an object actually gets used (Wilson & Kakkad, 1992).

As DRDBMSs perform more and more in-memory optimizations, they come closer to MMDBMSs. Depending on the dynamic use of data, good database management systems may detect that some data permanently resides in memory and use main memory database mechanisms to speed up operations for that data, whereas other parts of the database remain on disk and are treated as a conventional disk resident database. This allows having the best from both worlds.

### CONCLUSION

This article has presented the mechanisms of main memory databases that are used to improve performance, and it has presented their main differences to conventional databases. Nearly all parts of database management are affected due to the different properties of the underlying storage technology: Optimization algorithms, recovery, concurrency control, and backup all have to be tuned with the different characteristics in mind. On the other hand, main memory mechanisms are not limited to pure main-memory database management systems, but the optimizations used can be integrated into conventional databases to speed up parts of the operations performed.

## REFERENCES

- Blott, S., & Korth, H. F. (2002). An almost-serial protocol for transaction execution in main-memory database systems. In *Proceedings of the 28th VLDB Conference*.
- Bressoud, T. C., Clark, T., & Kan, T. (2001). The design and use of persistent memory on the DNCP hardware fault-tolerant platform. In *International Conference on Dependable Systems and Networks* (pp. 487-492). IEEE.
- DeWitt, D. J., Katz, R. H., Olken, F., Shapiro, L. D., Stobbraker, M. R., & Wood, D. (1984). Implementation techniques for main memory database systems. In *Proc. of the ACM SIGMOD Conf.* (pp. 1-8). ACM Press.
- Garcia-Molina, H., & Salem, K. (1987). High performance transaction processing with memory resident data. In *Proceedings of the International Workshop on High Performance Transaction Systems*.
- Garcia-Molina, H., & Salem, K. (1992). Main memory database systems: An overview. *IEEE Transactions on Knowledge and Data Engineering*, 4, 509-516.
- Lehman, T. J., & Carey, M. J. (1986). A study of index structures for main memory database management systems. In *Proceedings of the 12th International Conf. on Very Large Data Bases* (pp. 294-303). Morgan Kaufmann.
- Lin, J.-L., & Dunham, M. H. (1996). Segmented fuzzy checkpointing for main memory databases. In *Proceedings of the 1996 ACM Symposium on Applied Computing* (pp. 158-165). ACM Press.
- Mao, Y., Zhang, Y., Wang, D., & Zheng, W. (2002). LND: A reliable multi-tier storage device in NOW. *SIGOPS Operating Systems Review*, 36(1), 70-80.
- Meixner, M., & Buchmann, A. (2004). HADES—A highly available distributed main memory reliable storage. In *Proceedings of the 2004 High Performance Computing & Simulation (HPC&S) Conference* (pp. 50-56).
- Rao, J., & Ross, K. A. (1999). Cache conscious indexing for decision-support in main memory. *VLDB, Proceedings of the 25th International Conference on Very Large Databases* (pp. 78-89). San Francisco: Morgan Kaufman.
- Salem, K., & Garcia-Molina, H. (1989). Checkpointing memory-resident databases. In *Proceedings of the Fifth International Conference on Data Engineering* (pp. 452-462). IEEE Computer Society.
- Wedekind, H., & Zoerntlein, G. (1986). Prefetching in realtime database applications. In *Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data* (pp. 215-226). ACM Press.
- Wilson, P. R., & Kakkad, S. V. (1992). Pointer swizzling at page fault time: Efficiently and compatibly supporting huge address spaces on standard hardware. In *1992 International Workshop on Object Orientation and Operating Systems* (pp. 364-377). IEEE Computer Society Press.
- Woo, S.-K., Kim, M.-H., & Lee, Y.-J. (1997). Accommodating logical logging under fuzzy checkpointing in main memory databases. In *International Database Engineering and Application Symposium* (pp. 53-62). IEEE.

## KEY TERMS

**Abort:** Cancels all modifications of a transaction.

**ACID Properties:** Properties of transactions: atomicity, an operation is either completely performed or not at all; consistency, an operation transfers the database from one consistent state to another consistent state; isolation, intermediate states of a transaction are not visible to the outside; durability, changes made to a database are persistent.

**Cache:** Memory that mirrors often used parts of a slower but larger memory. The term cache mainly refers to the function not to the memory technology. Cache can be standard random access memory that is used to speed up disk accesses but it also can be very specialized high-speed memory that is used to speed up processor accesses to main memory.

**Commit:** Activates all modifications performed by a transaction, makes them visible to the outside, and makes all modifications durable.

**Concurrency Control:** The task of the concurrency control is to coordinate the concurrent execution of several transactions so that the chosen consistency properties (e.g., ACID properties) are not violated.

**Main Memory:** Memory that is used for storing data and program code and that can be directly accessed by the processor (random access memory).

**Recovery:** The task of recovery is to return the database to a consistent state after a crash: The effects of committed transactions must be guaranteed to be persistent, and effects of not committed transactions must be undone. Recovery requires that all modifications are written to some stable storage as part of a commit or else these modifications would be lost in a crash.

**Transaction:** Group of operations that are either performed all or none.

# Managing Inconsistent Databases Using Active Integrity Constraints

**Sergio Flesca**

*DEIS Università della Calabria, Italy*

**Sergio Greco**

*DEIS Università della Calabria, Italy*

**Ester Zumpano**

*DEIS Università della Calabria, Italy*

## INTRODUCTION

Integrity constraints are a fundamental part of a database schema. They are generally used to define constraints on data (functional dependencies, inclusion dependencies, exclusion dependencies, etc.), and their enforcement ensures a semantically correct state of a database. As the presence of data inconsistent with respect to integrity constraints is not unusual, its management plays a key role in all the areas in which duplicate or conflicting information is likely to occur, such as database integration, data warehousing, and federated databases (Bry, 1997; Lin, 1996; Subrahmanian, 1994). It is well known that the presence of inconsistent data can be managed by “repairing” the database, that is, by providing consistent databases, obtained by a minimal set of update operations on the inconsistent original environment, or by consistently answering queries posed over the inconsistent database.

The motivation of this work stems from the observation that in repairing a database it is natural to express among a set of update requirements the preferred ones, that is, those actions which, besides making the database consistent, also maintain the preferred information. The novelty of our approach consists in the formalization of *active integrity constraints*, a flexible and easy mechanism for specifying the preferred updates, that is, the actions that should be performed if an integrity constraint is not satisfied. In some sense, active integrity constraints represent a restricted form of active rules sufficient to (declaratively) express database repairs but without the typical problems of active databases, such as termination and procedural interpretation of rules. Thus, in the general case, active integrity constraints can be thought of as a means to define an “intended” repairing strategy.

Recently, there have been several proposals considering the computation of repairs and queries over

inconsistent databases (Arenas, Bertossi & Chomicki, 1999, 2000; Greco, Greco & Zumpano, 2001; Wijzen, 2003). Other works have investigated the updating of data and knowledge bases through the use of active rules and nonmonotonic formalisms. The application of the ECA (event-condition-action) paradigm of active databases to policies—collection of general principles specifying the desired behavior of systems—has been investigated in Chomicki, Lobo, and Naqvi (2003). In this work, the authors propose the introduction of active constraints to describe under which circumstances a set of actions cannot be executed simultaneously. In Alferes et al. (2000), the problem of updating knowledge bases represented by logic programs has been investigated. More specifically, the authors introduce the notion of updating a logic program  $P$  by means of another logic program  $U$  (denoted by  $P \oplus U$ ) and a new paradigm, called *dynamic logic programming*, to model dynamic program update. The new paradigm has been further investigated in Alferes et al. (2002), where the language LUPS (Language for Dynamic Updates), designed for specifying changes to logic programs, has been proposed.

Nonmonotonic formalisms, such as *Revision Programs* (Marek, Pivkina & Truszczyński, 1998; Marek & Truszczyński, 1998), are based on the extension of the logic programming paradigm. Unlike earlier approaches in belief revision, where updates were represented in classical theories, revision programs are a collection of rules with nonclassical semantics; they can be interpreted as inference rules and are used to update interpretations. ECA languages based on revision programs have been proposed as well in Baral (1997).

The approach here proposed differs both from ECA languages, as only sets of actions making the input database consistent are allowed, and from revision programs, as actions can be enforced not only by the initial state of the database.

## BACKGROUND

We assume that readers are familiar with relational and deductive databases (Abiteboul, Hull & Vianu, 1995; Ullman, 1988).

A (disjunctive Datalog) rule  $r$  is a clause of the form  $A_1 \vee \dots \vee A_k \leftarrow B_1, \dots, B_m, \text{not } B_{m+1}, \dots, \text{not } B_n$ ,  $k+m+n>0$

where  $A_1, \dots, A_k, B_1, \dots, B_n$  are atoms of the form  $p(t_1, \dots, t_h)$ ,  $p$  is a predicate symbol of arity  $h$ , and the terms  $t_1, \dots, t_h$  are constants or variables (Eiter et al., 1998). The disjunction  $A_1 \vee \dots \vee A_k$  is the *head* of  $r$ , while the conjunction  $B_1, \dots, B_m, \text{not } B_{m+1}, \dots, \text{not } B_n$  is the *body* of  $r$ . We also assume the existence of the binary built-in predicate symbols (comparison operators) which can only be used in the body of rules.

The *Herbrand Universe*  $U_p$  of a program  $P$  is the set of all constants appearing in  $P$ , and its *Herbrand Base*  $B_p$  is the set of all ground atoms constructed from the predicates appearing in  $P$  and the constants from  $U_p$ . A term, (resp. an atom, a literal, a rule, or a program) is *ground* if no variables occur in it. A rule  $r'$  is a *ground instance* of a rule  $r$ , if  $r'$  is obtained from  $r$  by replacing every variable in  $r$  with some constant in  $U_p$ . We denote by  $ground(P)$  the set of all ground instances of the rules in  $P$ . An interpretation of  $P$  is any subset of  $B_p$ .

An interpretation  $M$  for  $P$  is a model of  $P$  if  $M$  satisfies each rule in  $ground(P)$ . The (model-theoretic) semantics for a positive program, say  $P$ , assigns to  $P$  the set of its *minimal models*  $MM(P)$ , where a model  $M$  for  $P$  is minimal, if no proper subset of  $M$  is a model for  $P$  (Minker, 1982). The more general *disjunctive stable model semantics* also applies to programs with (unstratified) negation (Gelfond & Lifschitz, 1991). For any interpretation  $I$ , denote with  $P^I$  the ground positive program derived from  $ground(P)$  (1) by removing all rules that contain a negative literal *not a* in the body and  $a \in I$ , and (2) by removing all negative literals from the remaining rules. An interpretation  $M$  is a (disjunctive) stable model of  $P$  if and only if  $M \in MM(P^M)$ . For general  $P$ , the stable model semantics assigns to  $P$  the set  $SM(P)$  of its *stable models*. It is well known that stable models are minimal models (i.e.,  $SM(P) = MM(P)$ ) and that for negation free programs, minimal and stable model semantics coincide (i.e.,  $SM(P) = MM(P)$ ). Observe that stable models are minimal models which are “supported”; that is, their atoms can be derived from the program. An alternative semantics which overcomes some problems of stable model semantics has been recently proposed in Greco (1999).

## INTEGRITY CONSTRAINTS

Integrity constraints express semantic information over data, that is, relationships that must hold among data in the theory. Generally, integrity constraints, denoted as IC, represent the interaction among data and define properties which are supposed to be explicitly satisfied by all instances over a given database schema. Therefore, they are mainly used to validate database transactions.

### Definition 1

A full (or universal) integrity constraint is a formula of the first order predicate calculus of the form:

$$(\forall X) [ B_1 \wedge \dots \wedge B_n \wedge \phi \supset A_1 \vee \dots \vee A_m \vee \psi_1 \vee \dots \vee \psi_k ]$$

where  $A_1, \dots, A_m, B_1, \dots, B_n$  are base positive literals,  $\phi, \psi_1, \dots, \psi_k$  are built-in literals,  $X$  denotes the list of all variables appearing in  $B_1, \dots, B_n$  and it is supposed that variables appearing in  $A_1, \dots, A_m, \phi, \psi_1, \dots, \psi_k$  also appear in  $B_1, \dots, B_n$ .

In the definition above, the conjunction  $B_1 \wedge \dots \wedge B_n \wedge \phi$  is called the *body*, and the disjunction  $A_1 \vee \dots \vee A_m \vee \psi_1 \vee \dots \vee \psi_k$  the *head* of the integrity constraint. Moreover, an integrity constraint is said to be *positive* if no negated literals occur in it. Classical definitions of integrity constraints only consider positive nondisjunctive constraints, called *embedded dependencies* (Kanellakis, 1991).

Often we shall write our constraints in a different format by moving literals from the head to the body and vice versa.

## REPAIRING INCONSISTENT DATABASES

Intuitively, a *repair* for a (possibly inconsistent) database  $D$  is a minimal consistent set of insert and delete operations which makes  $D$  consistent, whereas a consistent answer for a query consists of two sets containing, respectively, the maximal set of true and undefined atoms which match the query goal; atoms which are neither true nor undefined can be assumed to be false.

More formally:



## Definition 2

Given a (possibly inconsistent) database  $D$ , a repair for  $D$  is a pair of sets of atoms  $(R^+, R^-)$  such that (1)  $R^+ \cap R^- = \emptyset$ , (2)  $D \cup R^+ - R^- \models IC$  and (2) there is no pair  $(S^+, S^-) \neq (R^+, R^-)$  such that  $R^+ \subset S^+$ ,  $R^- \subset S^-$  and  $D \cup S^+ - S^- \models IC$ . The database  $D \cup R^+ - R^-$  will be called the repaired database.

Thus repaired databases are consistent databases, derived from the source database by means of a minimal set of update operations. In more detail, for any repair  $R$  of a given database  $D$ ,  $R^+$  (resp.  $R^-$ ) denotes the set of tuples which will be added to (deleted from) the database. Observe that any repair  $R$  is a consistent set of update operations ( $R^+ \cup R^- = \emptyset$ ). In the following, for a given repair  $R$  and a database  $D$  with integrity constraints  $IC$ ,  $R(D) = D \cup R^+ - R^-$  denotes the application of  $R$  to  $D$ , whereas  $\mathbf{R}(D, IC)$  denotes the set of all possible repairs for  $D$ .

## Definition 3

Given a database  $D$  and a set of integrity constraints  $IC$ , an atom  $A$  is true (resp. false) with respect to  $(D, IC)$  if  $A$  belongs to all repaired databases (resp. there is no repaired database containing  $A$ ). The set of atoms which are neither true nor false are undefined.

Thus, true atoms appear in all repaired databases whereas undefined atoms appear in a proper subset of repaired databases. Given a database  $D$  and a set of integrity constraints  $IC$ , the application of  $IC$  to  $D$ , denoted by  $IC(D)$ , defines three distinct sets of atoms: the set of true atoms  $IC(D)^+$ , the set of undefined atoms  $IC(D)^u$ , and the set of false atoms  $IC(D)^-$ .

## ACTIVE INTEGRITY CONSTRAINTS

In this section, we present, in an informal way, an extension of integrity constraints that allows specification for each constraint the actions to be performed to satisfy it. The actions are defined by means of insertions and deletions.

## Definition 4

An update atom is either of the form  $+A$  or of the form  $-A$ , where  $A$  is a base atom. An active integrity constraint  $r$  is a formula of the first order predicate calculus of the form:

$$r = (\forall X) [\Phi \supset \Psi]$$

where  $\Phi$  is a range restricted conjunction of literals, and  $\Psi$  is a disjunction of update atoms.

Given an active integrity constraint  $(\forall X) [\Phi \supset \Psi]$ , we denote with  $St(r)$  the standard integrity constraint  $(\forall X) [\Phi \supset]$  derived from  $r$  by removing the head update atoms. Moreover, for a set of active integrity constraints  $AIC$ ,  $St(AIC)$  denotes the corresponding set of standard integrity constraints, that is,  $St(AIC) = \{St(r) \mid r \in AIC\}$ .

We start by defining the truth value of ground atoms and ground update atoms with respect to a database  $D$  and a consistent set of update atoms  $R$ , that is, a set not containing two update atoms of the form  $+A$  and  $-A$ . The truth value of built-in atoms and conjunctions is given in the standard way.

## Definition 5

Let  $D$  be a database and  $R$  a consistent set of update atoms for  $D$ , then a positive ground literal  $A$  is true in  $(D, R)$  if  $A \in R(D)$ , a negative ground literal  $not\ A$  is true in  $(D, R)$  if  $A \notin R(D)$ , a ground update atom  $+A$  (resp.  $-A$ ) is true in  $(D, R)$  if  $A \in R^+$  (resp.  $A \in R^-$ ).

In the following, given a database  $D$  and a consistent set of update atoms  $R$ , a ground active constraint  $\phi \supset \varphi$  is said to be (1) *true* w.r.t.  $(D, R)$  if  $\phi$  is true in  $(D, R)$  and (2) *applied* w.r.t.  $(D, R)$  if both  $\phi$  and  $\varphi$  are true in  $(D, R)$ .

A consistent set of update atoms  $R$  is a repair for a database  $D$  and a set of active integrity constraints  $AIC$  only if  $R$  is a repair for  $D$  and  $St(AIC)$ .

Every minimal set of update atoms  $R$  such that  $R(D) \models AIC$  is a repair for  $D$ ; moreover, not all repairs contain atoms which could be derived from the active integrity constraints.

## Definition 6

Let  $D$  be a database,  $AIC$  a set of active integrity constraints and  $R$  a repair for  $D$  and  $AIC$ . Then,  $R$  is said to be founded if for every ground update atom  $u \in R^+$  (resp.  $u \in R^-$ ), there is a ground active integrity constraint  $r: \phi \supset \varphi$  and an update atom  $+u$  (resp.  $-u$ )  $\in \varphi$  such that  $\phi$  is true in  $(D, R)$ .

Given a database  $D$  and a set of active integrity constraints  $AIC$ ,  $\mathbf{FR}(D, AIC)$  denotes the set of founded repairs for  $D$ .

## Example 3

Consider the integrity constraints:

$$\forall (E, P, D) [mgr(E, P) \wedge prj(P, D) \wedge not\ emp(E, D) \supset +emp(E, D)]$$



$$\forall(E1,E2,D,S) [\text{emp}(E,D1) \wedge \text{emp}(E,D2) \wedge D1 \neq D2 \supset -\text{emp}(E,D1) \vee -\text{emp}(E,D2)]$$

The first constraint states that every manager  $E$  of a project  $P$  carried out by a department  $D$  must be an employee of  $D$ , whereas the second one says that every employee must be in only one department.

Consider now the database  $D = \{ \text{mgr}(e1,p1), \text{prj}(p1,d1), \text{emp}(e1,d2) \}$ . There are three repairs for  $D$ :

- $R1 = \{ -\text{mgr}(e1,p1) \}$ ,
- $R2 = \{ -\text{prj}(p1,d1) \}$ , and
- $R3 = \{ +\text{emp}(e1,d1), -\text{emp}(e1,d2) \}$ .

$R3$  is the only founded repair as only the update atoms  $+\text{emp}(e1,d1)$  and  $-\text{emp}(e1,d2)$  are derivable from the active constraints.

From the previous considerations we have that  $\mathbf{FR}(D,AIC) \subseteq \mathbf{R}(D,AIC) \subseteq \mathbf{R}(D,St(AIC))$ .

## EXAMPLES

The use of active integrity constraints allows the specification of preference criteria; moreover, if a founded repair exists, it is ensured it has been obtained by only performing actions specified by the conditioned updates atoms in the head of the active integrity constraints. In this section, we show that the expression of preferences by means of active integrity constraints gives us the possibility of formulating powerful queries expressing hard problems.

### Example 4

- **Map coloring:** The following set of constraints AIC checks if the coloring of a (possibly partially colored) map can be completed by using only the two colors *red* and *blue*.

$$\forall(X,P) [\text{country}(X,P) \wedge \text{not col}(X,\text{red}) \wedge \text{not col}(X,\text{blue}) \wedge \text{not col}(X,\text{yellow}) \supset +\text{col}(X,\text{red}) \vee +\text{col}(X,\text{blue})]$$

$$\forall(X,Y,C) [\text{border}(X,Y) \wedge \text{col}(X,C) \wedge \text{col}(Y,C) \supset -\text{col}(X,C) \vee -\text{col}(Y,C)]$$

For each country, we know the name and the population (expressed in millions of inhabitants) while the relation *border* says whether two countries are neighbors. The two constraints state that colored nodes can be (re-)colored with one of two available colors.

Observe that in the above example, if we delete from the second constraint the update atoms in the head, as colored nodes cannot be recolored, the expressed problem consists in completing the coloring of the map.

In the following, we consider a graph  $G = \langle V, E \rangle$  defined by means of a unary predicate *node* and a binary predicate *edge*.

### Example 5

- **Clique:** A clique of a given graph  $G$  is a set of nodes such that every pair of nodes in it is connected.

$$\forall(X) [\text{node}(X) \wedge \text{not } c(X) \wedge \text{not } nc(X) \supset +c(X) \vee +nc(Y)]$$

$$\forall(X,Y) [c(X) \wedge c(Y) \wedge \text{not edge}(X,Y), \wedge X \neq Y \supset -c(X) \vee -c(Y)]$$

where  $c(x)$  means that the node  $x$  belongs to the clique, and  $nc(x)$  means that  $x$  does not belong to the clique. Initially, the relations  $c$  and  $nc$  are empty and the input database consists of nodes and edges.

### Example 6

- **Max Clique:** A clique of a given graph  $G$  is a set of nodes such that every pair of nodes in it is connected. A clique with maximum cardinality is called max-clique.

$$\forall(X) [\text{node}(X) \wedge \text{not } c(X) \wedge \text{not } nc(X) \supset +c(X)]$$

$$\forall(X,Y) [c(X) \wedge c(Y) \wedge \text{not edge}(X,Y), \wedge X \neq Y \supset -c(X) \vee -c(Y)]$$

where  $c(x)$  means that the node  $x$  belongs to the max-clique and  $nc(x)$  means that  $x$  does not belong to the max-clique. Initially, the relations  $c$  and  $nc$  are empty and the input database consists of nodes and edges.

## CONCLUSION AND FUTURE TRENDS

In this article, we have introduced *active integrity constraints*, a simple and powerful form of active rules with declarative semantics, well suited for computing database repairs and consistent answers. The problem with active integrity constraints is that the existence of founded repairs, in the general case, is not guaranteed.

Under the proposed semantics, called *prescriptive*, the allowed actions are exactly those specified by the constraints. Under such a semantics, the existence of

founded repairs, in the general case, is not guaranteed.

We are currently investigating a different type of semantics called *preferable* where actions are interpreted as preference conditions on the set of possible repairs, so that admitting repairs and consistent answers.

A general approach for the computation of repairs and consistent answers in the presence of databases with universal integrity constraints has been proposed in Greco et al. (2001). The technique, presented in Greco et al. (2001), can also be extended for dealing with active integrity constraints.

## REFERENCES

- Abiteboul, S., Hull, R., & Vianu, V. (1995). *Foundations of databases*. Addison-Wesley.
- Alferes, J.J., Leite, J.A., Pereira, L.M., Przymusinska, H.C., & Przymusinski T.C. (2000). Dynamic updates of non-monotonic knowledge bases. *Journal Logic Programming*, 45(1-3), 43-70.
- Alferes, J.J., Pereira, L.M., Przymusinska, H.C., & Przymusinski, T.C. (2002). LUPSA language for updating logic programs. *Artificial Intelligence*, 138(1-2), 87-116.
- Arenas, M., Bertossi, L., & Chomicki, J. (1999). Consistent query answers in inconsistent databases. *Proceedings of the International Conference on Principles of Database Systems* (pp. 68-79).
- Arenas, M., Bertossi, L., & Chomicki, J. (2000). Specifying and querying database repairs using logic programs with exceptions. *Proceedings of the International Conference on Flexible Query Answering* (pp. 27-41).
- Baral, C. (1997). Embedding revision programs in logic programming situation calculus. *IJCAI Conference*, 30(1), 83-97.
- Baral, C., & Zhang, Y. (2001). On the semantics of knowledge update. *IJCAI Conference* (pp. 97-102).
- Brewka, G., & Eiter, T. (1999). Preferred answer sets for extended logic programs. *Artificial Intelligence*, 109(1-2), 297-356.
- Bry, F. (1997). Query answering in information system with integrity constraints. *IFIP WG 11.5 Working Conference on Integrity and Control in Information Systems*.
- Chomicki, J., Lobo, J., & Naqvi, S.A. (2003). Conflict resolution using logic programming. *IEEE Transactions Knowledge Data Engineering*, 15(1), 24.
- Flesca, S., & Greco, S. (2001). Declarative semantics for active rules. *TPLP*, 1(1), 43-69.
- Gelfond, M., & Lifschitz, V. (1993). Representing action and change by logic programs. *J. Log. Program.*, 17(2/3/4), 301-321.
- Grant, J., & Subrahmanian, V.S. (1995). Reasoning in inconsistent knowledge bases. *IEEE Transactions on Knowledge and Data Engineering*, 7(1), 177-189.
- Greco, G., Greco, S., & Zumpano, E. (2001). A logic programming approach to the integration, repairing and querying of inconsistent databases. *Proceedings of the International Conference on Logic Programming*.
- Kanellakis, P.C. (1991). Elements of relational database theory. In J. van Leewen (Ed.), *Handbook of theoretical computer science, volume 2*. North-Holland.
- Lin, J. (1996). A semantics for reasoning consistently in the presence of inconsistency. *Artificial Intelligence*, 86(1), 75-95.
- Marek, V.W., Pivkina, I., & Truszczyński, M. (1998). Revision programming = logic programming + integrity constraints. *Computer Science Logic*, (pp. 73-89).
- Marek, V.W., & Truszczyński, M. (1998). Revision programming. *Theoretical Computer Science*, 190(2), 241-277.
- Sakama, C., & Inoue, K. (2000). Prioritized logic programming and its application to commonsense reasoning. *Artificial Intelligence*, 123(1-2), 185-222.
- Subrahmanian, V.S. (1994). Amalgamating knowledge bases. *Proceedings of the ACM ToDS*, 19(2), 291-331.
- Ullman, J.K. (1988). Principles of database and knowledge-base systems, 1.
- Wijsen, J. (2003). Condensed representation of database repairs for consistent query answering. *Proceedings of the ICDT*, 19(2), 378-393.

## KEY TERMS

**Active Integrity Constraint:** A formula of the first order predicate calculus of the form:  $r = (\forall X) [\Phi \supset \Psi]$  where  $\Phi$  is a range restricted conjunction of literals, and  $\Psi$  is a disjunction of update atoms.

**Consistent Answer:** A set of tuples, derived from the database, satisfying all integrity constraints.

**Consistent Database:** A database satisfying a set of integrity constraints.

## Managing Inconsistent Databases Using Active Integrity Constraints

**Data Integration:** A process providing a uniform integrated access to multiple heterogeneous information sources.

**Database Repair:** Minimal set of insert and delete operations which makes the database consistent.

**Disjunctive Datalog Program:** A set of rules of the form:

$$A_1 \vee \dots \vee A_k \leftarrow B_1, \dots, B_m, \text{not } B_{m+1}, \dots, \text{not } B_n,$$

$k+m+n > 0$

where  $A_1, \dots, A_k, B_1, \dots, B_n$  are atoms of the form  $p(t_1, \dots, t_h)$ ,  $p$  is a predicate symbol of arity  $h$  and the terms  $t_1, \dots, t_h$  are constants or variables.

**Integrity Constraints:** Set of constraints which must be satisfied by database instances.

**Inconsistent Database:** A database violating some integrity constraints.

# Mathematics of Generic Specifications for Model Management, I

Zinovy Diskin<sup>1</sup>

SWD Factory, Latvia and Universal Information Technology Consulting, USA

*This article (further referred to as Math-I), and the next one (further referred to as Math-II, see p. 359), form a mathematical companion to the article in this encyclopedia on Generic Model Management (further referred to as GenMMt, see p.258 ). Articles Math-I and II present the basics of the arrow diagram machinery that provides model management with truly generic specifications. Particularly, it allows us to build a generic pattern for heterogeneous data and schema transformation, which is presented in Math-II for the first time in the literature.*

## INTRODUCTION

Generic MMt (gMMt) is a novel view on metadata-centered problems manifested by Bernstein, Halevy, and Pottinger (2000). The main idea is to implement an environment where one could manipulate models as holistic entities irrespectively of their internal structure. It can be done only within an integral framework for abstract generic specifications of relations between models and operations with models. However, building truly generic specifications presents a problem of a new kind for the DB community, where such familiar tools as first-order logic or relational algebra cannot help. Amongst the most pressing gMMt problems listed in Bernstein (2003)—the most complete presentation of gMMt agenda to date—almost all are *specification* problems.

Fortunately, the community does not have to develop the desired framework from scratch. As is manifested in *GenMMt*, appropriate methodology and specification techniques are already developed in mathematical category theory (CT) and waiting to be adapted to gMMt needs. The goal of *Math-I* and *Math-II* is to outline foundations of the categorical approach to MMt and demonstrate how it works in a few simple examples.

Our plan is as follows. The next section of the article describes *Kleisly arrows*—the main vehicle of the approach. The section following it presents a simple example of model merge as a sample demonstrating how the categorical treatment of MMt procedures works. We begin by representing models in a graph-based format called *sketch* and describe a general pattern for sketch merging. After that, we reformulate the pattern in abstract terms to

make it applicable to any sort of models, not necessarily sketches, although models are still supposed to be similar (be instances of the same metamodel). The last section summarizes this work by describing a generic arrow framework for *homogeneous* MMt.

The next step—managing models’ heterogeneity, particularly data and schema translation—is a highly nontrivial issue, a truly generic solution to which is often considered to be impossible (Bernstein, 2003). Nevertheless, it does exist and is presented (for the first time in the literature) in *Math-II*. Based on some category theory ideas, data and schema translation is specified in an abstract generic way via the so-called *Grothendieck* mappings. After that, homogeneous MMt specifications presented in *Math-I* can be considered as *abbreviations* for heterogeneous specifications, where arrows are interpreted as Grothendieck mappings. Particularly, the abstract pattern of homogeneous model merge developed in *Math-I* can be applied for heterogeneous merge as well. The last section of *Math-II* summarizes the results and outlines a general mathematical framework underlying generic MMt specifications.

## MODEL MAPPINGS AS KLEISLY ARROWS

### Do Model Mappings Really Map?

In the literature, the term “model mapping” usually refers to a specification of correspondence between models. It was noted by many authors that a general format of such correspondence must involve derived items of the models in question. In recent surveys, such as Lenzerini (2002), it became common to describe a mapping between schemas  $S$  and  $T$  as a set of assertions of the type  $Q_S \rightsquigarrow Q_T$ , where  $Q_S$  and  $Q_T$  are some queries over schema  $S$  and  $T$ , respectively, and  $\rightsquigarrow$  denotes some comparison operator between query outputs. It is assumed that queries (operations) are terms formed by strings of operation symbols and variables, and assertions are formulas, i.e., strings composed from terms and logical operators.

This framework is typical for string-based algebra and logic familiar to the community, but it has a few inherited

drawbacks for generic MMT applications. First of all, string-based terms and formulas are inadequate for expressing queries and constraints over graph-based models like ER or UML diagrams. In addition, it is unclear how to define composition of mappings in this setting. Not surprisingly, in all concrete applications and implementations of this framework, it is specialized for working with relational models; see Fagin, Kolaitis, Miller, and Popa (2003); Madhavan and Halevy (2003); and Velegrakis, Miller, and Popa (2003).

A more general framework is proposed in Bernstein et al. (2000). The main idea is to reify mappings as models and to attach correspondence assertions to elements of these model mappings (see Figure 2 in *GenMMT*). In this way model mappings become spans (whose *legs*, i.e., projection arrows, are model morphisms), and their composition is defined as composition of spans. Though semantically clear, span composition is much more complicated than composition of arrows, and the problem of composing expressions still persists. A common drawback of both approaches is that queries appear as something foreign to the models and need a special interface (Problem 3 in Table 2 in *GenMMT*).

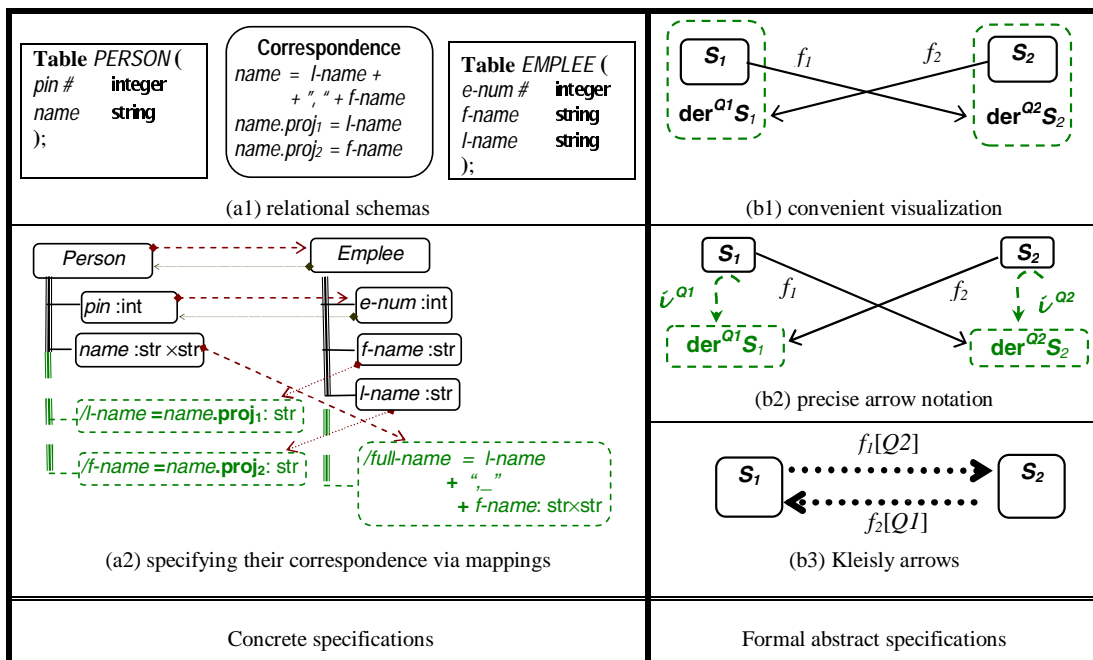
All these problems are eliminated as soon as we consider the issue in the generic framework of *categorical algebra* developed in CT in the 60<sup>th</sup>-70<sup>th</sup> (see Manes, 1976, for a basic account). The key observation is that operations/queries against the model can be denoted by adding new derived items to the model and labeling them

with the operation's name. Then correspondence between models is specified by mappings between models augmented with derived (computable) items. A simple example of how it works is presented in the left column of Figure 1. In the right column, Diagrams (b1), (b2) present an immediate abstraction of this description: Models (schemas)  $der^{Q1}S_1$ ,  $der^{Q2}S_2$  are augmentations of the original schemas with derived items produced by sets of queries  $Q1, Q2$ , respectively, and arrows  $i^{Q1}, i^{Q2}$  in Diagram (b2) denote inclusions of schemas.

An essential feature of the specifications in Figure 1 is that the arrows are *functional* mappings that send each item from the source to a single item in the target. We do not need to reify mappings by new intermediate models nor do we need to consider many-to-many relationships between models. Although reification of model correspondence by intermediate models (spans) is sometimes necessary, many MMT tasks can be specified with pure functional mappings (morphisms). From now on, we will use the term model mapping in the sense of functional mapping between models.

In our abstraction efforts in the right column, we can go even further and consider query expressions attributed to arrows rather than to models, as it is shown by Diagram (b3). In this way we come close to the notation with which we began our discussion, but note that in our case a figurative arrow  $S \rightsquigarrow T$  is just an abbreviation of pair  $(Q, f)$ , with  $Q$  a set of queries to the target schema and  $f: S \rightarrow der^{QT}$  a (functional) mapping of the source schema

Figure 1. Specifying correspondences between models via Kleisly morphisms





to the  $Q$ -augmentation of the target one. In CT, such arrows were called *Kleisly morphisms* after H. Kleisly, one of the pioneers of categorical algebra, and we will borrow this name from CT. By purely typographical reasons, Kleisly arrows in our figures are denoted by dotted-bold rather than zigzag arrows.

## Kleisly Operation

A key feature of Kleisly machinery is that mappings between models can be naturally extended to mappings between augmented models. Having a mapping  $f: S \rightarrow T$  and a set of queries  $Q$  over schema  $S$ , we expand  $f: S \rightarrow T$  to a mapping  $f^Q: \mathbf{der}^Q S \rightarrow \mathbf{der}^Q T$  by mapping each derived item  $y = \mathbf{q}(x_1, \dots, x_n)$  in  $\mathbf{der}^Q S$  ( $\mathbf{q}$  is a query from  $Q$  and  $x_i$  are its arguments) to an item  $\mathbf{q}(x_1 f, \dots, x_n f)$  in  $\mathbf{der}^Q T$ , where  $x.f$  denotes application of mapping  $f$  to  $x$ , that is, the  $f$ -image of  $x$ . In other words, we *homomorphically* map derived items in the source model to *similar* derived items in the target model. Further, we will call the procedure of extending a mapping to a homomorphism the *Kleisly operation*.

The composition of Kleisly morphisms  $(Q_1, f_1): S \rightsquigarrow T$  and  $(Q_2, f_2): T \rightsquigarrow U$  is easily defined as the ordinary functional composition

$$S \xrightarrow{f_1} \mathbf{der}^{Q_1} T \xrightarrow{f_2^{Q_1}} \mathbf{der}^{Q_1} \mathbf{der}^{Q_2} U = \mathbf{der}^{Q_1(Q_2)} U$$

where  $Q_1(Q_2)$  denotes the set of query expressions obtained by substituting queries from  $Q_2$  into queries from  $Q_1$ .<sup>2</sup> The problem of expression manipulation engine stated in Bernstein (2003) is now resolved: The job is done by the Kleisly operation and functional mapping compositions.

## Kleisly Arrows and Schema Equivalence

The Kleisly approach immediately reveals the role of derived items and queries in the issue of schema equivalence. Consider the example in Figure 1. Intuitively, it is clear that the two schemas present the same information; the only difference between them is in the choice of basic data from which other data is derived. This intuition can be formally explicated by building two mutually inverse mappings between schemas  $\mathbf{der}^{Q_1} S_1$  and  $\mathbf{der}^{Q_2} S_2$ . The mappings are built by consecutive application of Kleisly operations to the initial configuration of schemas and mappings and state a homomorphic (with respect to the operations involved) bijection between the schemas  $\mathbf{der}^{Q_1} S_1$  and  $\mathbf{der}^{Q_2} S_2$ . Thus, schemas  $\mathbf{der}^{Q_1} S_1$  and  $\mathbf{der}^{Q_2} S_2$  can be proven to be algebraically isomorphic with respect to some query language embodied into the notion of **der**-operator. In such cases we will say that schemas  $S_1$  and  $S_2$  are **der**-equivalent.

We will see in the next section that the notion of **der**-

equivalence is essential for schema integration: In fact, the result of schema merge is defined up to **der**-equivalence. Disregarding this peculiarity can lead to misunderstanding of the integration procedure. For example, in Buneman, Davidson, and Kosky (1992), they come to a strange conclusion that the merge operation is non-associative: In examples they consider,  $(S_1 + S_2) + S_3 \neq S_1 + (S_2 + S_3)$  (“+” denotes the merge operation). The examples are right but the conclusion is not: It can be shown that actually their models  $(S_1 + S_2) + S_3$  and  $S_1 + (S_2 + S_3)$  are **der**<sup>Q</sup>-equivalent for a set  $Q$  of few simple queries.

## MODEL MERGE CATEGORICALLY: A HOMOGENEOUS MMt SCENARIO VIA ARROWS AND DIAGRAM OPERATIONS

### Representing Models by Sketches

A vast body of work in schema integration was devoted to structural conflicts between schemas, when the same piece of reality is modeled differently in different schemas. Consider, for example, a situation when the same information is specified by an XML document and a UML diagram. Managing this diversity within syntactical frames is really hard if at all possible, and a shift to semantics is a must. The sketch format is a device for specifying semantics of a data schema in universal terms of sets and mappings irrespectively of the schema’s syntax (Diskin & Kadish, 2003). The sketch presentation of schemas provides homogeneity of their semantics and all types of structural conflicts are reduced to only one: Data considered basic in one schema are derived data in another schema (Cadish & Diskin, 1996).

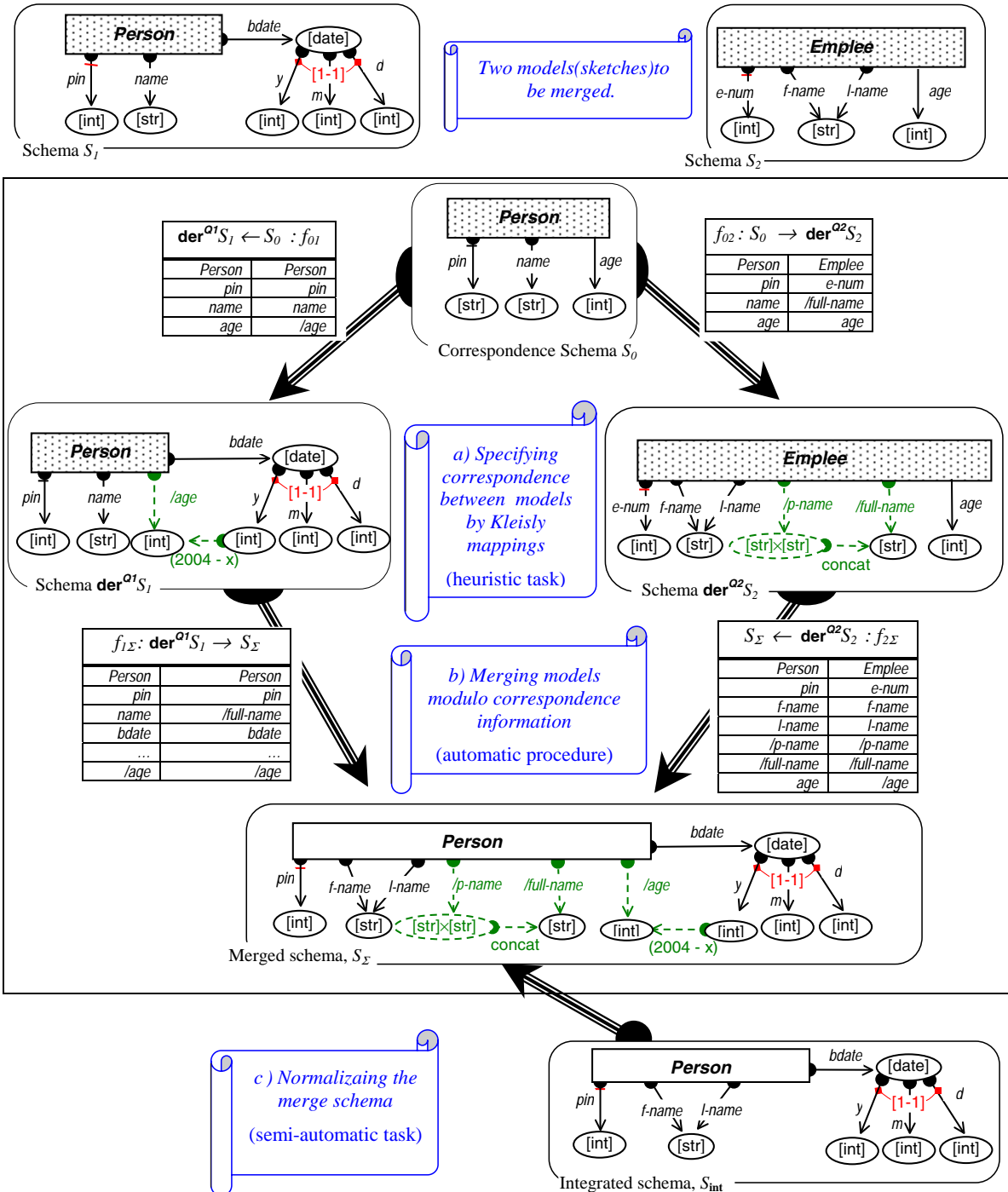
On the other hand, in many data models, representation of schemas by sketches is quite natural and easy. For example, let us consider the relational model. Semantically, rows of a relational table represent real-world objects while columns are mappings from the class of objects (rows) to value domains. From this viewpoint, it is reasonable to present a relational schema as a directed graph, whose nodes are object classes (table names) and value domains (SQL types), and arrows (columns) go from object classes to either value domains or other classes (foreign keys). For example, relational tables from Figure 2 in the previous article, GenMMt, are transformed into graphs in the upper part of Figure 2 below.

These graphs carry some additional structure visualized by markers hung on some nodes and arrows. The oval shape of a node is a visualization of declaring the node to be a value domain with predefined semantics; the latter is referred by the marker hung on the node. The bold

tail of an arrow declares the arrow to be a totally defined mapping. Marker [1-1] hung on an arrow multi-span diagram (see schema  $S_1$ ) means one-one correspondence between elements of the span's source ([bdate]-values) and tuples (triples [y,m,d]) of elements taken from the targets—one element for each leg of the span. A short bar

on an arrow's tail is the same [1-1] marker hung on a span consisting of a single arrow. A graph endowed with diagram predicate declarations (like just mentioned) is called a *sketch*. Thus, relational schemas are presented by sketches.

Figure 2. Model merge concretely (in the sketch representation of models)



## Merging Sketches

We specify the correspondence between schemas by Kleisly mappings between sketches. We first augment the original sketches with necessary derived items; they are shown on Figure 2 by dashed gray (green on a color display) lines.<sup>3</sup> Then we specify overlapping between schemas by introducing a new schema (sketch)  $S_0$  and specifying mappings from  $S_0$  into the augmented sketches (the second row of the figure).

The next step is entirely automatic. The augmented sketches are merged modulo the correspondence specified by the span  $S_0$ . Roughly, operation **merge** first forms a disjoint union of the sketches and then glues together items specified as equal by the span  $S_0$ . The result is a co-span: a sketch  $S_\Sigma$  together with two mappings into it,

$$f_{1\Sigma}: S_1 \rightarrow S_\Sigma \text{ and } f_{2\Sigma}: S_2 \rightarrow S_\Sigma$$

The merge sketch inevitably contains redundant derived items. To finish integration, it should be *normalized*, that is, rearranged to a form with a minimal redundancy. In other words, we need to find a subsketch  $S_{\text{int}} \subset S_\Sigma$  to be **der**-equivalent to  $S_\Sigma$ , i.e., such that  $S_\Sigma \subset \mathbf{der}^{Q_{\text{int}}} S_{\text{int}}$  for some set  $Q_{\text{int}}$  of operations (queries) over  $S_{\text{int}}$ . In the present case it is fairly easy: just take the basic (black) subsketch of  $S_\Sigma$ . In more complex situations, finding a suitable  $S_{\text{int}}$  can be not so trivial. Some general theory of normalizing sketches would be helpful and is still waiting for its development. Clearly, normalization can be automated but, in general, a human intervention is necessary.

## Towards Genericness Across Metamodels

Is the pattern of sketch merge applicable to other sorts of models? In other words, is it possible to define the merge operation in a generic way irrespectively of models' internal structure? Clearly, the key to the problem should be in semantic reformulation of merge, independent of syntactical peculiarities of the models.

Thinking semantically, given two schemas (models)  $S_1, S_2$ , we should define their merge as the least schema containing all the information specified by  $S_1$  and  $S_2$ . That is, the merged schema should be rich enough (but not more than necessary) so that by making appropriate queries to it we could recover data specified by schemas  $S_1$  and  $S_2$  and data that can be derived from them as well.

In other words, the notion of “all the information” specified by schemas assumes, first of all, that some query language (set of operations over data) QL is given. Then any schema  $S$  can be extended to a (huge) schema  $\mathbf{der}^{\text{QL}}S$  specifying all data derivable from data specified by  $S$  with

all possible queries against  $S$ . Now the merge of  $S_1, S_2$  is defined to be a schema  $S_\Sigma$  such that  $\mathbf{der}^{\text{QL}}S_\Sigma$  is the least schema to which schemas  $\mathbf{der}^{\text{QL}}S_1$  and  $\mathbf{der}^{\text{QL}}S_2$  could be mapped:

$$\mathbf{der}^{\text{QL}}S_1 \xrightarrow{f_{1\Sigma}} \mathbf{der}^{\text{QL}}S_\Sigma \xleftarrow{f_{2\Sigma}} \mathbf{der}^{\text{QL}}S_2$$

Of course, in concrete applications it is sufficient to deal with finite fragments of full **der**<sup>QL</sup>-completions, that is, schemas  $\mathbf{der}^Q S$  for some finite set  $Q$  of queries against  $S$ .

If the schemas overlap in information they specify, we need to refine this description. Overlapping is specified by some correspondence schema  $S_0$  encompassing the common items, and two mappings to show where these common items are placed in the schemas  $S_1, S_2$ , that is, by a span:

$$\mathbf{der}^{\text{QL}}S_1 \xleftarrow{f_{01}} S_0 \xrightarrow{f_{02}} \mathbf{der}^{\text{QL}}S_2$$

over the schemas to be related. Merge of  $S_1, S_2$  modulo this correspondence is a schema  $S_\Sigma$  as above and, in addition, arrow compositions  $f_{01} \gg f_{1\Sigma}$  and  $f_{02} \gg f_{2\Sigma}$  must be equal (see Cell (b) in Table 1).

The  $\mathbf{der}^{\text{QL}}S_\Sigma$ -schema's property of being “the least” can be explicated in the following (typical for CT) way. For any other schema  $T$  with mappings:

$$\mathbf{der}^{\text{QL}}S_1 \xrightarrow{f_{1T}} T \xleftarrow{f_{2T}} \mathbf{der}^{\text{QL}}S_2$$

such that compositions  $f_{01} \gg f_{1T}$  and  $f_{02} \gg f_{2T}$  are equal, there is a unique mapping:

$$!: \mathbf{der}^{\text{QL}}S_\Sigma \longrightarrow T$$

such that all the arrow diagrams involved are commutative. In this way we come to nothing but the definition of the join (push-out) operation well known in CT; see Table 2. In other words, it is reasonable to explicate the informal notion of schema merge (motivated by its practical applications and intuition) by the formal notion of join operation in a suitable category of schemas.

Now our procedure of merging sketches can be abstractly specified as shown in Cells (a), (b), and (c) of Table 1, where marker [join] on a square diagram refers to an abstract operation defined in Table 2. The rightmost cell presents an abbreviated specification via Kleisly arrows. Specifications in Table 1 are applicable to any category of models, not only sketches, where a Kleisly structure and the diagram operation **join** are defined.

Table 1. Model merge abstractly

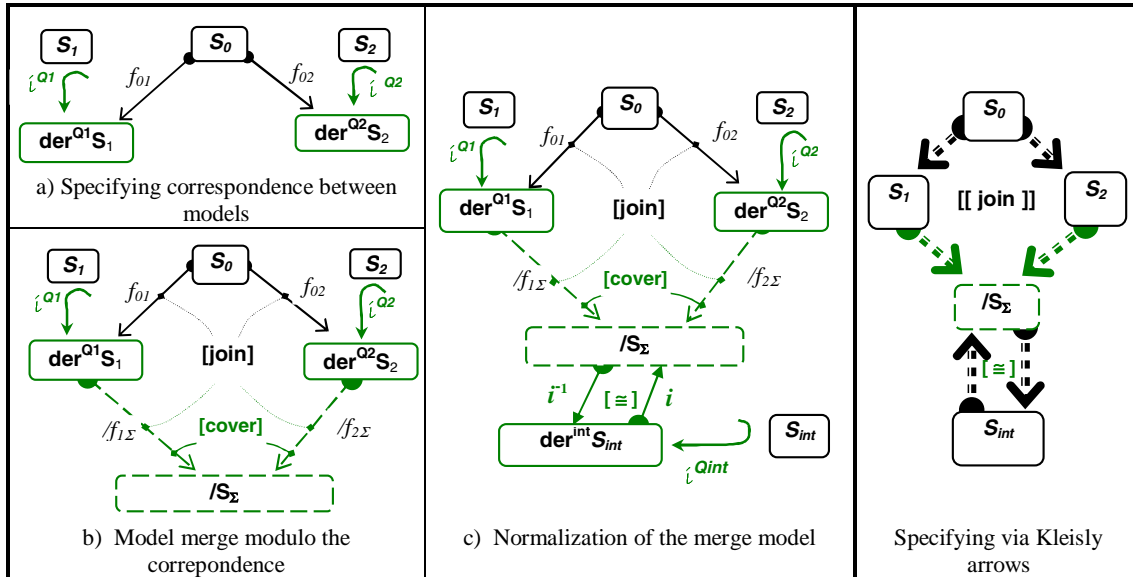


Table 2. A few basic diagram operations

Name and denotation	Input Diagram	Output Diagram	Semantics (informally)
Composition [comp]			Composition of mappings
Image [img]			A sort of image (range) of a mapping: $/I$ is the smallest subobject of $T$ (i.e., there 1-1 mapping from $/I$ into $T$ ) such that $f$ factors through it.
Push-out (categorical join) [join]			A sort of join (merge): $/J$ is the smallest object “containing” objects $T_1, T_2$ w.r.t. their “common” part $S$ . If $f_2$ is 1-1 mapping, then $/f_2^*$ is also 1-1 and $/J$ can be considered as $T_1$ augmented with the “extra”-part of $T_2$ w.r.t. $S$ .
Pull-back (categorical meet) [meet]			A sort of meet (intersection): $/M$ is the largest object “contained” in objects $T_1, T_2$ w.r.t. their “embedding” into $S$ . If $f_2$ is 1-1, then $/f_2^*$ is also 1-1 and $/M$ can be considered as the <i>coimage</i> of $T_2$ under $f_1$ , i.e., the largest subobject of $T_1$ whose image under $f_1$ is inside $T_2$ .
Difference [diff]			$/D$ is the smallest subobject of $S$ , whose (categorical) union with $T$ gives $S$ .

## MATHEMATICS OF HOMOGENEOUS GENERIC SPECIFICATIONS

Various metadata management tasks may require various operations with models. For example, we may need to extract the common part of two submodels of a given model (intersection) or to form a submodel complementing some given submodel to the entire model. Likewise, given a model mapping  $m: S \rightarrow T$ , we may need to have  $m$ -image of some submodel of  $S$  or  $m$ -coimage of  $T$ 's submodel. In general, MMT may require model counterparts of many familiar operations with sets and mappings (functions).

Specifying such operations for a particular type of model (relational schemas, XML DTDs, sketches) would not be a serious problem. However, for generic MMT we need to specify these operations in a metamodel independent way, that is, independently of syntactic peculiarities of models whatever they might be. At first glance, the problem looks unapproachable and the few attempts published in the literature tried to avoid it by using “cumulative generalization” rather than abstraction. The idea was to catalogue the constructs used in all the dialects of a modeling language and try to extract a smaller generating subset from them. Examples are “the most general” ER format by Atzeni and Torlone (1996) or “the most general” metamodel for OO modeling defined in the Object Management Group's (2002) Meta-Object Facility specification. Clearly, this sort of genericness is very restrictive and for model translation does not work at all.

A proper approach to the problem was found in category theory. The latter demonstrated that familiar operations with sets and mappings can be abstractly reformulated for objects of arbitrary nature in terms of arrows (mappings), without any references to objects' internal structure. A few examples of such a reformulation are presented in Table 2. Basically, they follow the pattern we have used in the previous section for abstract reformulation of model merge.

One can note that an essential part of these formulations is a requirement of being the smallest/largest object satisfying certain properties. Such a requirement is called a *universal property* and can be also formulated via arrows similarly to how we described the minimality of model merge in the previous section. A surprising result discovered in category theory is that there is a small basic set of universal operations from which an extremely rich set of other diagram operations can be generated by composition. Categories (universes of objects and mappings) possessing this set of operations are called *toposes*, and they are fairly remarkable. Namely, every constructive operation over sets and mappings, which one normally enjoys in a set universe, can be defined in an arbitrary topos as well.<sup>4</sup>

Thus, given a collection  $\mathcal{M}$  of similar models and mappings, if we have implemented a few diagram operations constituting  $\mathcal{M}$  as a topos, then by composing these operations we can also perform any constructive operation with models and mappings. Of course, this nice picture needs some reservations and details, and we will return to it in the last section of *Math-II*. First, however, we need to figure out whether this framework is also suitable for heterogeneous MMT—so far, our models were supposed to be similar. A priori, it is not clear how to manage a key MMT operation of data and schema translation in a generic way. This is the goal of the next article.

## REFERENCES

- Atzeni, P., & Torlone, R. (1996). Management of multiple models in an extensible database design tool. In *Lecture notes in computer science: Vol. 1057. International Conference on Extending Database Technology, EDBT'96* (pp. 79-95). Springer Verlag.
- Bernstein, P. (2003). Applying model management to classical metadata problems. In *International Conference on Innovative Database Research, CIDR'2003* (pp. 209-220).
- Bernstein, P., Halevy, A., & Pottinger, R. (2000). A vision for management of complex models. *SIGMOD Record*, 29(4), 55-63.
- Buneman, P., Davidson, S., & Kosky, A. (1992). Theoretical aspects of schema merging. In *Lecture notes in computer science: Vol. 580. International Conference on Extending Database Technology, EDBT'92*. Springer.
- Cadish, B., & Diskin, Z. (1996). Heterogeneous view integration via sketches and equations. In *Lecture notes in artificial intelligence: Vol. 1079. Foundations of Intelligent Systems, Ninth International Symposium, ISMIS'96* (pp. 603-612). Springer Verlag.
- Diskin, Z., & Kadish, B. (1997). A graphical yet formalized framework for specifying view systems. In *Advances in Databases and Information Systems, ADBIS'97: Vol. 1. Regular papers* (pp. 123-132). St. Petersburg, Russia: Nevsky Dialect.
- Fagin, R., Kolaitis, P., Miller, R., & Popa, L. (2003). Data exchange: Semantics and query answering. In *International Conference on Database Theory, ICDT'2003* (pp. 207-224).
- Lenzerini, M. (2002). Data integration: A theoretical perspective. In *ACM Symposium on Principles of Database Systems, PODS'2002* (pp. 233-246).



Manes, E. (1976). *Graduate texts in mathematics: Vol. 26. Algebraic theories*. Springer-Verlag.

Object Management Group. (2002). *Meta-Object Facility (MOF) specification, Version 1.4*. Retrieved from <http://www.omg.org/technology/documents/formal/mof.htm>

Velegarakis, Y., Miller, R., & Popa, L. (2003). Mapping adaptation under evolving schemas. In *International Conference on Very Large Databases, VLDB'2003*.

## KEY TERMS

**Category Theory (CT):** A branch of modern algebra, providing a language and machinery for defining and manipulating mathematical structures in an abstract and generic way. The method of CT is to present the universe of discourse as a collection of objects (nodes) and morphisms (arrows) between them. The latter can be composed and so the universe is presented as a **category**: a directed graph with composable arrows. For example, models of a given sort (metamodel) and mappings between them form a category in a very natural way. If one wants to work with a heterogeneous universe of models (that is, models of different metamodels), one should use **fibrations** described in *Math-II*.

The philosophy of CT is that everything one wants to say about objects, one must say in terms of arrows between objects. Correspondingly, an object's structure is a structure over its arrow interface. This way of defining structures and manipulations with them is often called **arrow thinking**.

**Diagram Operation:** An operation over objects and morphisms that takes a certain configuration of them as an input, *input diagram*, and augments it with a few new—derived—objects and morphisms matching the shape of the *output diagram*. Table 2 presents a few examples: Derived items are shown with dashed gray (green on a color display) lines and their names are prefixed by slash (UML's notation).

**Diagram Predicate:** A property of a certain configuration, *diagram*, of objects and morphisms.

**Kleisly Morphism:** A basic construct of categorical algebra. In the context of MMT, a Kleisly morphism from model  $S$  to model  $T$  is a mapping of the source model's items to derived or basic items of the target model. In other words, a Kleisly morphism assumes (i) a set of queries  $Q$  to the target model and (ii) a mapping (function)  $m: S \rightarrow \mathbf{der}^Q T$  into the augmentation of the target model with derived items corresponding to the queries from  $Q$ .

**Sketch:** A basic construct of categorical logic. Roughly, it is a directed graph in which some diagrams are marked with predicate symbols taken from a predefined signature. A sketch morphism is a mapping of the underlying graphs compatible with marked diagrams. A fact of fundamental importance for generic MMT is that any metamodel (relational, XML Schema, a specific sort of ER or UML diagram) can be specified by a sketch. Then the corresponding universe of models (relational schemas, XML DTDs, ER diagrams) can be presented as the category of instances of the corresponding sketch. It allows applying many CT ideas and results to MMT.

## ENDNOTES

- <sup>1</sup> Partially supported by Grant 05.1526 from the Latvian Council of Science.
- <sup>2</sup> To make this description precise, we need a formal and generic notion of query, that is, an operation over data specified by a schema. Such a notion was defined in Diskin and Kadish (1997).
- <sup>3</sup> Derived items have to be added to sketches only within a strict discipline of applying diagram operations. For example, in sketch  $\mathbf{der}^{Q_2} S_2$ , the operation **pair** takes two arrows  $fname$ ,  $lname$  as input and produces an arrow  $pname$  (paired-name). Then composition of arrows  $pname$  and  $concat$  results in an arrow  $fullname$ . For sketch  $\mathbf{der}^{Q_1} S_1$ , arrow  $age$  is defined as composition of arrows  $bdate \gg y \gg (2004 - *)$ . To distinguish between basic and derived items, the latter are shown in dashed gray (green on a color display) lines to suggest that extent of these items is computable and does not need to be stored. In addition, names of derived items are prefixed with slash, as suggested by UML.
- <sup>4</sup> However, in a topos these operations have unusual properties. For example, if  $T$  is a subobject of object  $S$ , and  $S-T$  is (a sort of) their difference (see Table 2), intersection of  $T$  and  $S-T$  is not necessarily empty. And this is just what we have in MMT with graph representation of models: Subgraphs  $T$  and  $S-T$  do not have common arrows but can have common nodes. The logic/algebra of toposes can be precisely specified and turned out to be an algebraic version of the so-called intuitionistic higher-order predicate logic rather than classical two-valued first-order logic.

# Mathematics of Generic Specifications for Model Management, II

Zinovy Diskin<sup>1</sup>

SWD Factory, Latvia and Universal Information Technology Consulting, USA

*This article (further referred to as Math-II), and the previous one (further referred to as Math-I, see p. 351), form a mathematical companion to Generic Model Management (further referred to as GenMMt, see p. 258). While Math-I is dealing with homogeneous MMt, the goal of Math-II is to develop machinery for heterogeneous MMt, where models are not assumed to be similar.*

## INTRODUCTION

The main issue of heterogeneous MMt is data and model transformation from one format (metamodel) into another. A general setting for the task is as follows: Data (instance)  $D_1$  over a schema  $S_1$  in metamodel  $M_1$  needs to be transformed into data  $D_2$  over a schema  $S_2$  in metamodel  $M_2$ . So far, a generic format for the task has not been described in the literature. However, the problem can be solved in the arrow framework: Table 1 presents the entire operation as a sequence of standard diagram operations with data, models, metamodels, and mappings between them. The diagrams in the table look simple, but they are actually abbreviations of more complex constructs. To understand these abbreviations, we will first consider a simple example of the task and then abstract it to a generic specification.

## GETTING STARTED: AN EXAMPLE

Suppose we need to translate data stored in the graphic sketch format into relational tables. This task is reasonable only if the relational schema elements—tables, columns, and primary and foreign keys—are somehow implicitly present in the sketch format elements—nodes, arrows, and diagram predicates. To perform the task, we need to make this presence explicit and build a mapping from the metaschema of relational schemas into the metaschema of sketches.

For this purpose, we need to specify the metaschemas in some common format. For instance, we could specify them as relational schemas (then data schemas will be treated as relations) or sketches (then data schemas will be sketch instances, i.e., special graphs). In our example

we chose sketches; this format is universal and convenient for presentation. Thus, we need to build a sketch  $M_{rel}$  specifying the relational metaschema; a sketch  $M_{ske}$  specifying the metaschema of sketches; and a sketch mapping (Kleisly arrow)  $m: M_{rel} \rightarrow \mathbf{der}^Q M_{ske}$  interpreting relational constructs by either basic or derived constructs of the sketch format.

To simplify presentation, we will consider a simple version of relational schemas, where the primary key consists of only one column. This version is specified by sketch  $M_{rel}$  in the lower right cell of Figure 1 (for a while, do not pay attention to the right parts of nodes' and arrows' names following after colons). An instance of this sketch as a data schema is a graph  $S$  labeled by  $M_{rel}$ 's items in such a way that the labels set a graph mappings  $s: S \rightarrow M_{rel}$ . An example is presented in the cell just above the  $M_{rel}$  cell. In that graph,  $/S^*$ , objects' names (identifiers) are prefixed with # while values are braced with “”; names of arrows are omitted. Nodes' and arrows' labels follow their names after the colons and are violet on a color display. Such labeled graphs are, in fact, graph mappings as shown by the bold dotted arrow near the left border of the cell. The source of such a mapping is the graph itself while the labels determine the mapping into the graph pointed by the bold dotted arrow. It is easy to see that mapping a graph into the relational metamodel sketch  $M_{rel}$  makes it a relational schema. For example, the labeled graph  $/S^*$  in Figure 1 is nothing but the relational schema specified in the cell on the right. These two cells present the same metadata.

A sketch specifying an essentially simplified sketch metamodel,  $M_{ske}$ , is shown in the left lower cell of the figure (do not consider dashed items for a moment). This metamodel says that a sketch consists of *Node* objects and *Arrow* objects, and each *Arrow* is assigned with two *Nodes* called its *source* and *target*. The class of nodes is partitioned into *objectClasses* and *valueDomains*; the former have *names* (that are *strings*, i.e., elements of the domain [str]), while the latter have *SQL types*. In addition, each *objectClass* node is assigned an arrow called the *key* to the class, and the subgraph consisting of these *key* arrows must be acyclic (constraint **KGA: Key (sub)Graph is Acyclic**).<sup>2</sup> An instance of this sketch is a labeled graph, that is, a graph mapping  $\sigma: S \rightarrow M_{ske}$ , like that one shown in the cell just above the metaschema  $M_{ske}$ . More compact

presentation of this instance is shown on the left.

For interpreting items of the relational metaschema, sketch  $M_{\text{ske}}$  must be augmented with the corresponding derived items. They are shown by dashed gray (green on a color display) lines and defined as specified by Figure 1A; their names are prefixed with slash, as suggested by UML. Now we can explicate correspondence between sketch and relational schemas by building an interpretation mapping  $m: M_{\text{rel}} \rightarrow \mathbf{der}^Q M_{\text{ske}}$ . This mapping is specified in Figure 1 by labeling the  $M_{\text{rel}}$ 's items by  $M_{\text{ske}}$ 's items; labels stand after colon in  $M_{\text{rel}}$ -items' names and are shown in violet on a color display.

## DATA TRANSLATION IS DETERMINED BY METAMODEL INTERPRETATION

We will see how metamodel interpretation governs the entire transformation procedure. To illustrate the idea, we apply the general procedure to a very simple piece of data shown in the left-most upper cell of Figure 1.

The first step is to present data as a graph mapping  $\delta: D \rightarrow S$  from the data graph to the schema graph, and the schema (data on the metalevel) as a graph mapping  $\sigma: S \rightarrow M$  into the metaschema graph. These mappings, specified via labeling (as described above), are shown in the left column half of Figure 1 (by the abuse of notation, we use labels of schema arrows as their “names” for labeling the data-graph arrows). Identifiers for the four data links in the leftmost top cell are denoted in graph  $D$  by  $\#P_i$ ,  $i=1, \dots, 4$ ).

Definitions of derived items augmenting the metaschema are query specifications for its instances—schemas. Executing these queries augments the schema with derived items shown in the dashed lines, which, in their turn, are query specifications for data and propagate to augmenting data. In this way we obtain a chain of extended mappings  $\overline{D} \xrightarrow{\overline{\delta}} \overline{S} \xrightarrow{\overline{\sigma}} \overline{M} = \overline{M_{\text{ske}}}$ , where overlined capital letters denote augmented graphs. To simplify notation, further in the article we will always interpret arrows by Kleisly mappings and omit bars over objects' and arrows' names. In other words, we will work in a Kleisly category over graphs (determined by a choice of legitimate operations/queries with data; see *Math-I*).

The next step is to select that part of the schema sketch  $S$ , which is mapped by  $\sigma$  into the range of the interpretation  $m$ , and rearrange its labeling to sketch  $M_{\text{rel}}$ . This is done by applying to the span  $S \xrightarrow{\sigma} M_{\text{ske}} \xleftarrow{m} M_{\text{rel}}$  the diagram operation of **meet** explained in Table 2 of *Math-I*. The result is a new graph  $/S^*$  together with two mappings:

vertical,  $/\sigma^*: /S^* \rightarrow M_{\text{rel}}$ , and horizontal,  $S \xleftarrow{/m^*} /S^*$ .<sup>3</sup>

The vertical mapping is shown in the right middle cell of Figure 1; it makes  $/S^*$  a relational schema. The horizontal mapping (informally described by the visual graphic similarity between  $/S^*$  and  $S$ ) relates  $/S^*$  with the original schema sketch and shows how the labels are changed.

Finally, we select that part of the data graph  $D$ , which is mapped by  $\delta$  into the range of the schema mapping  $/m^*$ , and rearrange its labeling to graph  $/D^*$ . This is again done by applying the **meet**-operation and produces a new graph  $/D^*$  together with two mappings:

vertical,  $/\delta^*: /D^* \rightarrow /S^*$ , and horizontal,  $D \xleftarrow{/m^{**}} /D^*$ .

The former mapping makes data graph  $/D^*$  an instance over schema  $/S^*$ , and the latter relates it to the original data graph.

Properties of the interpretation  $m$  determine properties of the entire procedure. Since in our example  $m$  is injective (one-one) and the **meet**-operation preserves injectivity, objects  $/S^*$  and  $/D^*$  are, in fact, sub-objects of their sketch-format counterparts. In fact, we did nothing but select these sub-objects and rearrange their labeling. Thus, for one-one mapping  $m$ , **meet** works with labels of data items rather than with data itself.

Another special property of our example is that the schema sketch  $S$  is so simple that the range of mapping  $\sigma$  comes to be inside of the range of mapping  $m$ , i.e., inside of the image of  $M_{\text{rel}}$  under  $m$ . Hence, no data item was lost during the transformation. In general, the full sketch format is richer than relational, and some part of the full sketch metamodel goes beyond the range of  $m$ . If a piece of data,  $D$ , has a schema whose image under mapping  $\sigma$  goes beyond the range of  $m$ , then some data items from  $D$  will be lost during the transformation.

Considering this simple example in abstract terms leads to a generic pattern of data and schema transformation. It is described in the next section.

## DATA TRANSFORMATION VIA DIAGRAM CHASING

Table 1 presents an abstract generic specification of data transformation in the general setting described at the beginning of the article. The input information for the entire procedure is formed by (1) a data instance  $D_1$  over a source schema  $S_1$  and (2) two Kleisly morphisms: metaschema interpretation  $m_{12}: M_2 \rightsquigarrow M_1$  and schema mapping (view)  $s_{12}: S_2 \rightsquigarrow /S_1^*$  of the target schema into the  $M_2$  translation of the source schema. All the rest is done automatically by consecutive application of the two ge-



Table 1. Generic format of data transformation

Heterogeneous MMt		Homogeneous MMt	
Step 1	Step 2	Step 3	Step 4
<p>Specifying 1st half of input data for the operation: metaschema interpretation (Kleisly arrow) <math>m_{12}</math></p>	<p>Translating schema <math>S_1</math> into schema <math>S_1^*</math> in metamodel <math>M_2</math> and then rearranging the data <math>D_1</math> to data <math>/D_1^*</math> over schema <math>/S_1^*</math></p>	<p>Entering 2nd half of input data for the operation: schema <math>S_2</math> in metamodel <math>M_2</math> together with a view (Kleisly arrow) <math>s_{12}</math> to schema <math>/S_1^*</math></p>	<p>Rearranging data <math>/D_1^*</math> into data <math>/D_1^{**}</math> over schema <math>S_2</math>.</p>
<p><b>Quasi-homogeneous MMt:</b> Overall specification of Steps 1..4 via Grothendieck arrows and operations with them</p>			

neric diagram operations: Kleisly expansion for vertical arrows (see section “Kleisly Operation” in *Math-I*) and **meet** for the left-lower arrow spans (Table 2 in *Math-I*).

In more detail, after we have translated data and schema over metamodel  $M_1$  into metamodel  $M_2$  (see Steps 1 and 2 in Table 1), we can safely specify the target schema  $S_2$  as a view to the schema  $/S_1^*$ , that is, as a (homogeneous) Kleisly mapping  $s_{12}: S_2 \rightsquigarrow /S_1^*$  (Step 3 in Table 1). Computing data  $D_2$  is now nothing but computing a (materialized) view of data  $D_1$  specified by schema  $S_2$ . This computation is performed by applying the diagram operation **meet** once again (Step 4).

Note, the metaschema interpretation  $m_{12}$  sets the base of the entire procedure and in much determines its properties. Strictly speaking, we cannot even talk about schema  $S_2$  as a view to schema  $S_1$  until  $S_1$  is translated to a form similar to  $S_2$  (to  $S_2$  metaschema’s terms). On the other hand, we can introduce a new sort of arrows between schemas, say,  $f: S \rightrightarrows T$ , which would be a shorthand for pairs  $(m, s)$  with  $m: M_S \rightsquigarrow M_T$ , a Kleisly arrow (itself a shorthand!) between metaschemas and  $s: S \rightarrow /T^*$ , a homogeneous schema mapping. The target of this mapping,  $/T^*$ , is the translation of schema  $T$  into the  $M_S$  terms according to interpretation  $m$  as described above. We will call such arrows *Grothendieck morphisms* or *mappings*, after Alexander Grothendieck, the inventor of this fundamental categorical construct.

Grothendieck mappings provide a very convenient abbreviation. With Grothendieck arrows we can specify heterogeneous MMt procedures as if we are living in a homogeneous model world. For example, the task of data transformation looks like a simple view computation; see the lower half of Table 1. For another example, heterogeneous model merge can be specified as shown in the right-most column of Table 1 in *Math-I* but with Kleisly arrows replaced by Grothendieck arrows. Implementation, however, needs de-abbreviating the Grothendieck arrows and operations with them into arrows between model representations and operations over them, as shown in the upper half of Table 1 here.

### CATEGORICAL FOUNDATIONS FOR GENERIC MMt SPECIFICATIONS

Consider specifications in Table 1 of *Math-I* and Table 1 in this article once again. We have directed graphs, whose nodes denote data and schema representations, and arrows are mappings between them. The graphs are endowed with marked diagrams denoting either operations with representations and their mappings, for example, [meet] and [join], or properties of their configurations, for example, in cell (b) of Table 1 in *Math-I*, the pair of arrows  $/f_{1\Sigma}, /f_{2\Sigma}$  is declared to be *covering* as specified by the marker [cover]. Another example of a diagram



Table 2. Abstract MMt in categorical light

	Elementary Units	Repository Structure	Elementary Query	A Complete Query Language
Data Management	Value	Set of relations (tables)	Relational operation	Relational algebra (a version of first-order predicate calculus)
Model Management	Model, mapping	Sketch of models and mappings	Diagram operation	MMt doctrine/topos (a version of higher-order intuitionistic predicate calculus)

property is commutativity of all arrow squares involved (by default). Thus, our meta-specifications are sketches in some signature of diagram operations and predicates.

This format is generic across metamodels simply because metamodels are included into specifications, either explicitly, as in the upper part of Table 1, or implicitly, when we interpret arrows by Grothendieck mappings. The format is also generic across applications: Other MMt tasks may require other diagram operations and predicates to be applied, but the very pattern remains the same. Since operations involved should inevitably include arrow composition, the sketches we discuss are finite presentations of *categories* endowed with an extra structure.

This extra structure consists of two components. The first is a Kleisly structure,  $\mathbf{Q}$ . It is formed by query operations of augmenting nodes (representations) with new items and the corresponding expansions of arrows (see section “Model Mappings as Kleisly Arrows” in *Math-I*). The second component is an algebra  $\mathbf{A}$  of diagram operations (over representations and their mappings as holistic entities); see Table 2 in *Math-I* for examples. Using the notion of Kleisly mapping, we can “hide” the structure  $\mathbf{Q}$  inside of the arrows and consider diagram operations over the Kleisly category  $\mathbf{R}[\mathbf{Q}]$ . Thus, the MMt universe can be considered as a pair  $(\mathbf{R}[\mathbf{Q}], \mathbf{A})$  with  $\mathbf{R}[\mathbf{Q}]$  a category of model representations and Kleisly mappings between them, and  $\mathbf{A}$  a diagram algebra over it.

A question of fundamental importance for gMMt is whether there exists a set of diagram operations whose compositions can generate any MMt operation of practical interest (of course, here we do not consider heuristic MMt operations like schema matching). This sort of problem was studied in Category Theory in detail, and roughly the results are as follows.

Endowing a category  $\mathbf{C}$  with a certain set of diagram operations  $\mathbf{A}$  allows interpreting logical calculi in  $\mathbf{C}$ , and the other way round, a pair  $(\mathbf{C}, \mathbf{A})$  generates a certain logical calculus (the so-called internal language of  $\mathbf{C}$ ). In this way a category endowed with an extra algebraic structure and a logical calculus turn out to be basically equivalent constructs.<sup>4</sup> That is why pairs  $(\mathbf{C}, \mathbf{A})$  are called (*logical*) *doctrines* in categorical logic.

An important aspect of the notion of doctrine is that

the component  $\mathbf{A}$  actually refers to the entire pool of diagram operations legitimate over  $\mathbf{C}$  rather than to a specific set of basic operations generating  $\mathbf{A}$  by composition. Such pools of operations closed under composition are sometimes called *clones*. Thus, the notion of doctrine refers to a clone of diagram operations rather than to a specific set of operations generating it.<sup>5</sup>

A fine hierarchy of doctrines and their calculus counterparts was built in Category Theory. In the top part of the hierarchy are algebraically and logically rich categories called *toposes*, whose logical calculus counterparts are higher-order predicate logics, type theories, and constructive set theories. Roughly, in a topos one can perform all constructive operations with objects which one would normally enjoy with ordinary sets; particularly, take intersections and unions of objects, form their images and co-images under given morphisms, consider graphs (binary relations) of morphisms, take transitive closures of binary relations, and much more. Surprisingly, this extremely rich pool of manipulations with objects is generated by compositions of only three basic diagram operations: *arrow composition*, *meet*, and *powerset* (building the object of all sub-objects of a given object).

The category of graphs is proven to be a topos, and hence, every constructive operation with graphs can be presented as a composition of three basic topos operations. However, the *powerset* operation is extremely non-effective and, hence, should be excluded from the clone of MMt operations  $\mathbf{A}_{MMt}$ . Thus, the latter is a proper subclone of the clone of topos operations  $\mathbf{A}_{Top}$ ,  $\mathbf{A}_{MMt} \subset \mathbf{A}_{Top}$ . Since *powerset* is excluded from  $\mathbf{A}_{MMt}$ , we face the problem of finding an appropriate set of basic operations generating the entire clone  $\mathbf{A}_{MMt}$ . This set should include *image*, *meet*, *join* (see Table 2 in *Math-I*), some restricted version of *powerset* (discussed in Diskin & Kadish, 2003), and, perhaps, a few other important operations, which could be derived (in a topos) with the powerset operation but without it must be included into the basic set. Finding the latter is currently an open question. Nevertheless, there should be an object  $(\mathbf{C}_{MMt}, \mathbf{A}_{MMt})$  in the hierarchy of doctrines—let us call it *MMt doctrine* or *MMt topos*—in which algebra  $\mathbf{A}$  consists of effective diagram operations covering all gMMt’s needs.<sup>6</sup>



Figure 1. Example of data translation: sketches into relational tables

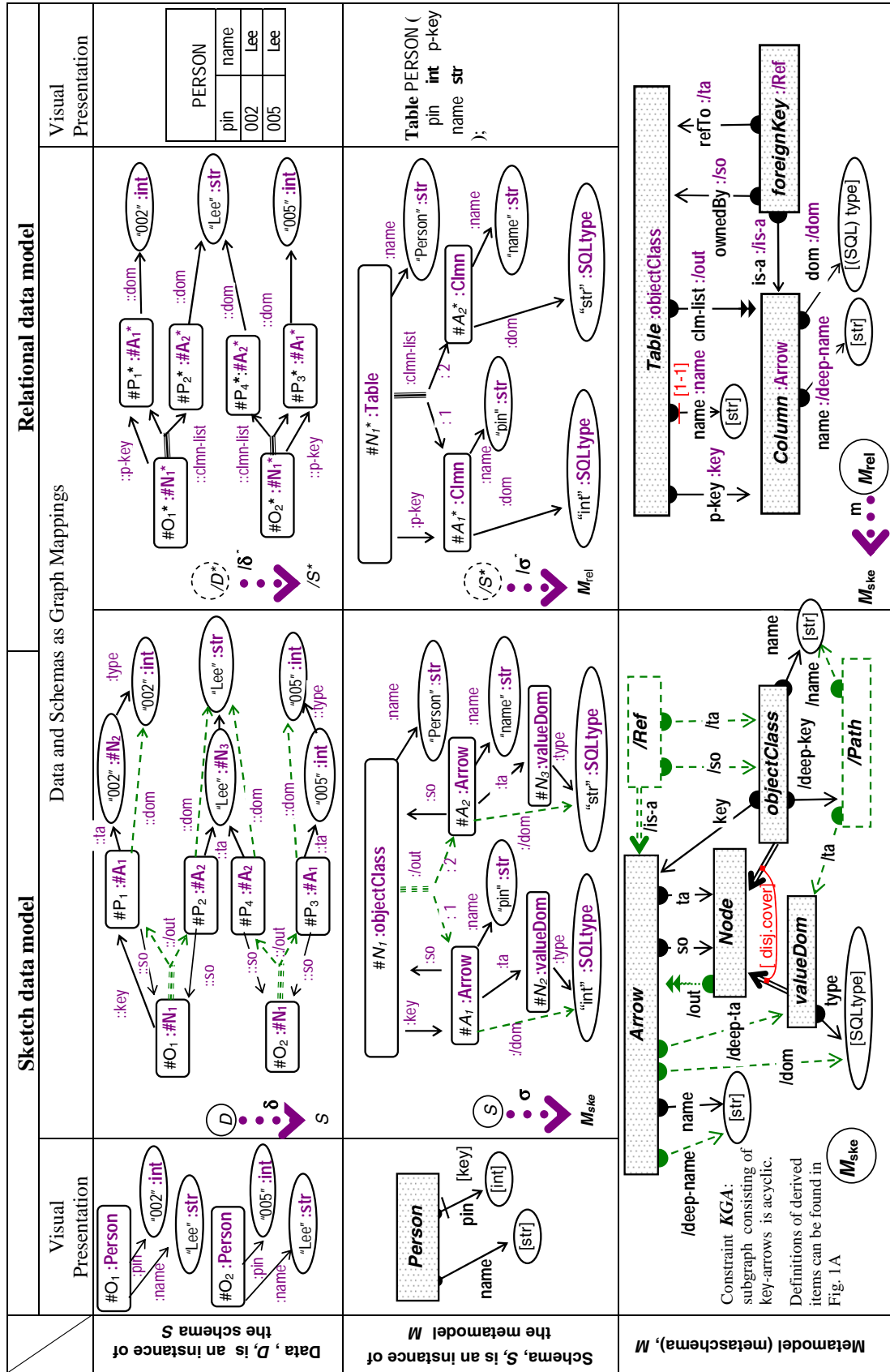


Figure 1A. Set  $Q = \{Q1, \dots, Q9\}$  queries specifying derived items of sketch  $M_{ske}$  in Figure 1. An extent of item  $X$  (a node or an arrow) is denoted by  $[[X]]$  (a set or a mapping resp.)

		The intended extents of derived nodes are defined as follows:
(Q1)	$[[/Ref]]$	$\stackrel{def}{=} \{A \in [[Arrow]] \mid A.[[ta]] \in [[objectClass]]\}$
(Q2)	$[[/Path]]$	$\stackrel{def}{=} \{(A_1, \dots, A_n), A_i \in [[Arrow]] : A_1.[[so]] \in [[objectClass]], A_i.[[ta]] = A_{i+1}.[[so]], A_n.[[ta]] \in [[valueDom]]\}$
		The extents of derived arrows are defined as follows: for any $A \in [[Arrow]], P \in [[Path]], N \in [[Node]], C \in [[objectClass]]$
(Q3)	$P.[[ta]]$	$\stackrel{def}{=} A_n.[[ta]]$
(Q4)	$P.[/name]$	$\stackrel{def}{=} "A_1.[[name]]\_ " A_2.[[name]]\_ \dots\_ " A_n.[[name]]\{"}$
(Q5)	$N.[/out]$	$\stackrel{def}{=} \text{the list of all } (A_1, \dots, A_k) \text{ with } A_i.[[so]] = N,$
(Q6)	$C.[/deep-key]$	$\stackrel{def}{=} \text{CASE } \{ \begin{array}{l} (C.[[key]]) \text{ if } C.[[key]].[[ta]] \in [[valueDom]]; \\ (C.[[key]] \mathbf{append} C.[[key]].[[ta]].[/deep\_key]) \text{ if } C.[[key]].[[ta]] \in [[objectClass]] \end{array} \},^*)$
(Q7)	$A.[/deep-ta]$	$\stackrel{def}{=} \text{CASE } \{ \begin{array}{l} A.[[ta]] \text{ if } A.[[ta]] \in [[valueDom]]; \\ A.[[ta]].[/deep-key].[/ta] \text{ if } A.[[ta]] \in [[objectClass]] \end{array} \},$
(Q8)	$A.[/dom]$	$\stackrel{def}{=} A.[/deep-ta].[[type]]$
(Q9)	$A.[/deep-name]$	$\stackrel{def}{=} \text{CASE } \{ \begin{array}{l} A.[[name]] \text{ if } A.[[ta]] \in [[valueDom]]; \\ "A.[[name]]\_ " A.[[ta]].[/deep-key].[/name]\{" \text{ if } A.[[ta]] \in [[objectClass]] \end{array} \}$
		<small>*) It's a recursive definition. It provides a well-defined result owing to the constraint <b>KGA</b> and finiteness of the graph</small>

Now we can fill in our Table 1 of abstract MMT's goals from *GenMMt* as shown in Table 2.

## CONCLUSION

Software engineering (like any other engineering field) is governed mainly by technological and economical reasons near the ground, rather than criteria of intellectual beauty and elegance in the sky. Nevertheless, it appears that for MMT, abstract compact specifications of high generality are “a matter of life and death rather than disposable luxury”.<sup>7</sup> Thus, it is nothing surprising in that the 30-year history of managing metadata has led to the “abstract-intensive” discipline of *generic MMT*.

A similar need in genericness emerged in mathematics in the 1940s, after many years of playing with mathematical structures, their mappings and transformations, and Category Theory was born. It offered a language and machinery for generic structure engineering, and one might expect that it could be useful for gMMt as well. Nevertheless, even with these considerations in mind, it still seems mysterious that constructs and ideas developed in an entirely abstract mathematical discipline appear as if specially designed to manage practical problems. On the other hand, many pieces of categorical

methods and constructions have already appeared in MMT literature and implemented in software independently of Category Theory (CT). Though they are often implicit and partially corrupted and not mathematically clean, it would not be an overstatement to say that today software engineers and researchers are rediscovering categorical specification patterns in an industry-adaptable form.

Having precise mathematical counterparts of generic MMT specifications would help MMT greatly. Moreover, what gMMt could borrow from CT does not amount to a few constructs and methods. In more than 50 years of work in pure mathematics and applications, CT has developed a special intuition and consistent methodology for specifying, reasoning, and operating complex structures in a generic way, which is often called *arrow* thinking. What gMMt should borrow from CT first of all is the arrow style of thinking, after which categorical methods and machineries could be properly adapted and applied to MMT problems.<sup>8</sup>

## REFERENCES

Barr, M., & Wells, C. (1995). *Category theory for computing science*. Prentice Hall International.

Diskin, Z. (1998). *Representing objects by values: Towards semantic foundations for object-oriented relational database systems*. (Tech. Rep. FIS-LDBD-9802). Frame Inform Systems, Riga, Latvia.

Diskin, Z. (2001). On modeling, mathematics, category theory and RM-ODP. In H. Kilov (Ed.), *WOODPECKER'2001: Workshop on Open Distributed Processing: Enterprise, computation, knowledge, engineering and realization* (pp. 38-54). Lisbon, Portugal: ICEIS Press.

Diskin, Z., & Kadish, B. (2003). Variable set semantics for keyed generalized sketches: Formal semantics for object identity and abstract syntax for conceptual modeling. *Data & Knowledge Engineering*, 47, 1-59.

Freyd, P., & Scedrov, A. (1990). *Categories, Allegories*. Elsevier Science Publishers. North Holland.

Goguen, J. (1991). A categorical manifesto. *Mathematical Structures in Computer Science*, 1(1), 9-67.

Jacobs, B. (1999). *Categorical logic and type theory*. Elsevier Science Publishers. North Holland.

Lawvere, F., & Schanuel, S. (1997). *Conceptual mathematics: A first introduction to categories*. Cambridge University Press. UK.

Poya, W. (1992). *An introduction to fibrations, topos theory, the effective topos and modest sets*. (Research Rep. ECS-LFCS-92-208). University of Edinburgh, School of Informatics, Laboratory for Foundations of Computer Science. Edinburgh, UK

**Fibration:** A basic construct of category theory. Roughly, it is a triple  $(\mathbf{C}, \mathbf{B}, \mathbf{f})$  with  $\mathbf{C}$  a category (of, say, models),  $\mathbf{B}$  another category (of metamodels) called *base*, and  $\mathbf{f}: \mathbf{C} \rightarrow \mathbf{B}$  a functor (mapping) between categories (that assigns to each model its metamodel). Given a base object (metamodel)  $M \in \mathbf{B}$ , those  $\mathbf{C}$ -objects (models, schemas)  $S$  for which  $S.\mathbf{f} = M$  and those  $\mathbf{C}$ -arrows between them  $s: S \rightarrow S'$  for which  $s.\mathbf{f} = \mathbf{id}_M$  (identity mapping of  $M$ ), form a category (of models in the metamodel  $M$ ),  $\mathbf{C}_M$ . This category is called the *fibre* over  $M$ . Fibres are mutually connected in the following way. Let  $m: M' \rightarrow M$  be a base arrow (metamodel interpretation) and  $S$  be an object over  $M$ ,  $S.\mathbf{f} = M$ . Then there is an object  $S^*$  over  $M'$ ,  $S^*.\mathbf{f} = M'$ , and an arrow  $m^*: S^* \rightarrow S$  over  $m$ ,  $m^*.\mathbf{f} = m$ , having some remarkable properties (their formulation is too technical to be presented here). In the MMt context, object  $S^*$  is the same model  $S$ , but its elements are renamed in terms of metamodel  $M'$  as governed by the interpretation  $m$ , and  $m^*$  is the renaming mapping. The technical properties just mentioned abstract this intuition in terms of morphisms in  $\mathbf{C}$  and  $\mathbf{B}$ . A basic account of fibrations can be found in Poya (1992) or Jacobs (1999); the former is much more lighter technically.<sup>9</sup>

**Topos:** A category endowed with a certain collection of diagram operations that allow one to perform many manipulations with objects and morphisms. Roughly, in a topos one can perform all constructive operations with objects: intersections, unions, and other analogs of common set theoretical operations, including the powerset (forming the object of all sub-objects of a given object). Typical examples of toposes are the category of sets and mappings between them or the category of graphs and graph mappings.

## KEY TERMS

**Doctrine:** A category endowed with a certain collection of diagram operations that allow one to perform certain logical manipulations with objects and morphisms. For example, in a category called *logos*, one can join and meet objects, take images and co-images of objects under given morphisms, and consider graphs (binary relations) of morphisms. In a *transitive* logos, one can, in addition, consider transitive closures of binary relations. A detailed account of doctrines can be found in Freyd and Scedrov (1990). Like relational algebra is an algebraic counterpart of some version of first-order predicate calculus, a categorical doctrine is a diagram-algebraic counterpart of some sort of logical calculus. Correspondingly, a hierarchy of logical calculi ranging from propositional to first-order predicate to higher-order predicate calculi gives rise to the corresponding hierarchy of categories—doctrines.

## ENDNOTES

- 1 Partially supported by Grant 05.1526 from the Latvian Council of Science.
- 2 Details about well-formed keyed sketches and the role of the **KGA** condition can be found in Diskin (1998); see also Diskin and Kadish (2003).
- 3 The choice of names for the internal items of graph  $/S^*$  is arbitrary, e.g., add \*-superscript to the names used in  $S$ .
- 4 One result of this kind is actually well known to the database community: Relational algebra is an algebraic equivalent of a version of first-order predicate calculus.
- 5 For example, the familiar notion of “relational algebra” refers to a certain well-known clone of operations over relations rather than to some specific set of basic operations generating it.

- <sup>6</sup> We might call this structure *weak topos*, or *quasi-topos*, or, better, *effective topos*, but unfortunately all these names are already used in CT. To avoid collision, naming this structure *MMt topos* or *MMt doctrine* seems to be reasonable, at least until its place in the hierarchy of doctrines will be precisely defined.
- <sup>7</sup> E. W. Dijkstra said these words about elegance in computing on another occasion, but they are just to the point for gMMt.
- <sup>8</sup> As a rule, formal mathematical constructs appear within consistent *systems* of concepts rather than discretely. Such systems normally explicate some integral non-trivial intuition, partially embedded in each of the constructs that the system consist of. This way the intuition behind a mathematical construct (like the roots of a tree) can go much wider and deeper than can be seen in the construct as such. The result is that in mathematical modeling, an apt formal counterpart,  $F$ , of a real world phenomenon/artifact,  $W$ , can offer much more than initially expected from stating the correspondence “ $F$  models  $W$ ”. The history of applying mathematics to science and engineering is full of examples when formalisms turned out to be surprisingly clever in their modeling and predictive capabilities. Category Theory, saturated with structures mutually modeling each other so that “everything is interpreted
- in everything”, is especially rich in such far-reaching integral intuitions. As a result, categorical reformulations of particular applied phenomena/artifacts can be very productive and helpful. A discussion of this phenomenon can be found in Goguen (1991) and Diskin(2001).
- <sup>9</sup> **About CT Books:** Category Theory is overly abstract even by mathematical standards, during CT’s early days, its internal nickname was “the abstract nonsense.” Though CT is extremely rich in ideas on structure engineering, their rigorous presentation inevitably involves a lot of technical details. Therefore, the authors of a CT book for nonmathematicians need to pass between *Scylla* of being comprehensible but limited, and *Charybdis* of being rich in material but overly technical. So far, it seems that no author team sailed safely between the monsters. For example, Lawvere and Schanuel (1997) are quite readable but cover only the basics. On the other hand, Barr and Wells (1995) describe all the constructs we consider in *Math-I* and *Math-II* but the text seems to be too complicated, even though the book is written with a bias towards CT applications in computer science. Hopefully, the MMt interpretation of CT constructs, provided in *Math-I* and *II*, will make the reader’s journey through CT seas more enjoyable.

# Metric Databases

**Edgar Chávez**

*Universidad Michoacana, Mexico*

**Gonzalo Navarro**

*University of Chile, Chile*

## INTRODUCTION

The advent of the digital age creates an increasing interest to search for information in large, unstructured repositories containing textual and multimedia data. Retrieval from those repositories requires more powerful query languages, which exceed the querying capabilities of traditional database technologies. The metric space model, described herein, is an extension of the exact searching paradigm aiming to cope with the new requirements.

Traditional databases are built around the concept of exact searching. The database is divided into *records*, each record having a fully comparable *key*. Queries to the database return all the records whose keys match the search key *exactly*. More sophisticated searches, such as range queries on numerical keys or prefix searching on alphabetical keys, still rely on the concept of exact comparison. Recently, database manager systems (DBMS) incorporate the ability of storing so-called multimedia objects, which nevertheless can rarely be searched by content.

The most distinguishing feature of multimedia objects is that there is no point in comparing them by exact equality. Rather, users are interested in the similarity between two objects. The metric space model gives a theoretical foundation to define a meaningful way to retrieve multimedia data. A *metric database* is a collection of digital objects (of any kind) with a *perceived* similarity and a formal way to compute this perceived similarity as a metric. The perceived similarity is provided by everyday experience or by experts.

The theoretical foundations for metric databases are well established. However, metric databases are not yet mature. The end users do not have the equivalent of a full DBMS for multimedia indexing. There is, however, at least one rather robust, open source implementation of an index for metric spaces (Ciaccia, Patella, & Zezula, 1997).

## APPLICATIONS OF METRIC DATABASES

Although retrieving multimedia data is one of the main applications for metric databases, their applications extend well beyond it. The following sections describe several of those applications (Chávez, Navarro, Baeza-Yates, & Marroquin, 2001).

### Query by Content in Multimedia Objects

New data types such as images, fingerprints, audio and video (i.e., multimedia data types) cannot be meaningfully queried in the classical sense. Not only cannot they be ordered, but they cannot even be compared by equality. No application will be interested in searching an audio segment exactly equal to a given one. The probability that two different images are pixel-wise equal is negligible unless they are digital copies of the same source. In multimedia applications, all the queries ask for objects similar to a given one. Some example applications are face recognition, fingerprint matching, voice recognition, and image retrieval.

These approaches are based on the definition of a similarity function among objects. Those functions are provided by an expert, but they pose no assumptions on the type of queries that can be answered. In many cases, the distance is obtained via a set of  $k$  features that are extracted from the object (e.g., in an image, a useful feature is the color histogram). Then each object is represented as its  $k$  features, that is, a point in a  $k$ -dimensional space.

### Information Retrieval

Although not considered a multimedia data type, unstructured text retrieval poses problems similar to mul-



timedia retrieval. This is because textual documents are in general not structured to easily provide the desired information. Text documents may be searched for strings that are present or not, but in many cases they are searched for semantic concepts.

The problem is solved by retrieving documents similar to a given query. The user can even present a document as a query, so that the system finds similar documents. Some similarity approaches are based on mapping a document to a vector of integer values, so that each dimension is a vocabulary word and the number of times the word appears in the document is the coordinate of the document in that dimension. Similarity functions are then defined in that space. Notice, however, that the dimensionality of the space is very high (thousands of dimensions).

## Text Retrieval

Another problem related to text retrieval is spelling. Because huge text databases with low quality control are emerging (e.g., the Web), and typing, spelling, or OCR (optical character recognition) errors are commonplace in the text and the query, documents that contain a misspelled word are no longer retrievable by a correctly written query. Models of similarity among words exist (variants of the “edit distance”) that capture those kinds of errors. In this case, one gives a word and wants to retrieve the words close to it.

## Computational Biology

ADN and protein sequences are the basic objects of study in molecular biology. Because they can be modeled as text, there is the problem of finding a given sequence of characters inside a longer sequence. However, an exact match is unlikely, and computational biologists are more interested in finding parts of a longer sequence that are similar to a given short sequence. That the search is not exact is because of minor differences in the genetic streams that describe sequences of similar functionality, evolution, experimental errors, and so on. The measure of similarity used is related to the type of differences one is interested in.

## Pattern Recognition and Function Approximation

A simplified definition of pattern recognition is the construction of a function approximator. One has a finite sample of the data, and each data sample is labeled as belonging to a certain class. When a fresh data sample

is provided, one must label this new sample as belonging to one of the known classes.

If the objects are  $m$ -dimensional vectors of real numbers, then a natural choice is neural nets and fuzzy function approximators. There is also another popular universal function approximator: the  $k$  nearest neighbor classifier. It consists of finding the  $k$  nearest objects to the unlabeled sample and assigning to this object the label having majority among the  $k$  nearest neighbors. Opposed to neural nets and fuzzy classifiers, the  $k$  nearest neighbor rule has zero training time, but if no indexing algorithm is used, it has linear complexity. Other applications of this universal function approximator are density estimation and reinforcement learning. In general, in any application in which one wants to infer a function based on a finite set of samples, there is a potential application.

## Audio and Video Compression

Audio and video transmission over a narrow-band channel is a serious problem in, for example, Internet-based audio and video conferencing. A frame (i.e., a static picture in a video, or a fragment of the audio) can be thought of as formed by a number of (possibly overlapped) subframes (e.g.,  $16 \times 16$  subimages in a video). In a very succinct description, the problem can be solved by sending the first frame as is and, for the next frames, sending only the subframes having a large difference from the previously sent subframes. This description encompasses the MPEG standard. This poses the need to efficiently find subframes similar to the one that is to be sent among a repository of recently sent subframes.

## THE METRIC SPACE MODEL

Proximity queries are those extensions of the exact searching from which one wants to retrieve objects from a database that are *close* to a given query object (Chávez, Navarro, Baeza-Yates, & Marroquin, 2001). The query object is not necessarily a database element. The concept can be formalized using the metric space model, in which a distance function  $d(x,y)$  is defined for every site in a set  $X$ . The distance function  $d$  has *metric* properties, that is, it satisfies  $d(x,y) \geq 0$  (positiveness),  $d(x,y) = d(y,x)$  (symmetry),  $d(x,y) = 0$  iff  $x = y$  (strict positiveness), and  $d(x,y) \leq d(x,z) + d(z,y)$  (triangle inequality). The latter is the key property permitting solutions better than brute force.

The database is a set  $U \subseteq X$ , and the query object,  $q$ , is an arbitrary element of  $X$ . A similarity query can be of two basic types: A *metric range query*  $(q,r)d = \{u \in U:$

$d(q,u) \leq r$ }, which retrieves all database elements at distance at most  $r$  to  $q$ ; and a  $k$  nearest neighbor query  $nn_k(q) = \{u_i \in U: \forall v \in Y, d(q,u_i) \leq d(q,v) \text{ and } |\{v_i\}| = k\}$ , which retrieves the  $k$  database elements closest to  $q$ .

Any such query can be answered by comparing the query  $q$  against the whole database, that is, computing  $d(q,u)$  for every  $u \in U$ . Because this is prohibitive, indexing techniques have been proposed to save as many distance computations as possible. Depending on the application, attention must also be paid to I/O costs and other CPU costs beyond computing distances.

There are some proximity searching problems in which a distance cannot be defined. Instead, a (dis)similarity is defined, which is a function not obeying the triangle inequality. It is common that the dissimilarity obeys a relaxed version of the triangle inequality, namely  $d(x,y) \leq v(d(x,z) + d(z,y))$ , with  $v \geq 1$  and with probability  $\epsilon > 0$ . Those cases can be dealt with by variants of the techniques used for metric spaces. If we cannot have even the relaxed version of the triangle inequality, then no better-than-brute-force approaches can be used.

### THE CURSE OF DIMENSIONALITY

Vector spaces, typically using Euclidean distance, can be seen as metric spaces where objects are arrays of numbers. There exist many indexing mechanisms specifically aimed at indexing vector spaces (Gaede & Günther, 1998). In general, these mechanisms obtain better per-

formance than general metric space indexes, especially when the space has few coordinates. However, the situation gets more complicated with higher dimensional spaces (Böhm, Berchtold, & Keim, 2001).

It is well known that the performance of any indexing algorithm on a uniformly distributed vector space will suffer from an exponential dependency on the dimension of the space. A random query with nonempty outcome will force any index to scan the whole database, rendering it useless. This is called the “curse of dimensionality,” and it makes uniformly distributed spaces of dimensions 20 or more, intractable in practice. On the other hand, if the data points are not uniformly distributed in the space, then some indexes are able to perform better by mapping the space to fewer dimensions without significantly losing distance information. The reason is that the *representational dimension* of the space (i.e., the number of coordinates) can be much larger than the *intrinsic dimension*, which is, intuitively, the lowest dimension from which the data points could be properly represented.

One of the benefits of seeing a vector space as a metric space is that the performance of the indexes directly depends on the intrinsic rather than the representational dimension, without need of any mapping. Even in metric spaces, where there is no notion of coordinates, there are spaces that are intrinsically more difficult to search than others, independent of the indexing method used. In order to accurately predict the achievable performance of searching a given metric space and to apply the most adequate index given its intrinsic difficulty, it is important to be able to quantify

Figure 1. A low-dimensional (left) and high-dimensional (right) histogram of distances, showing that on high dimensions virtually all the elements become candidates for exhaustive evaluation

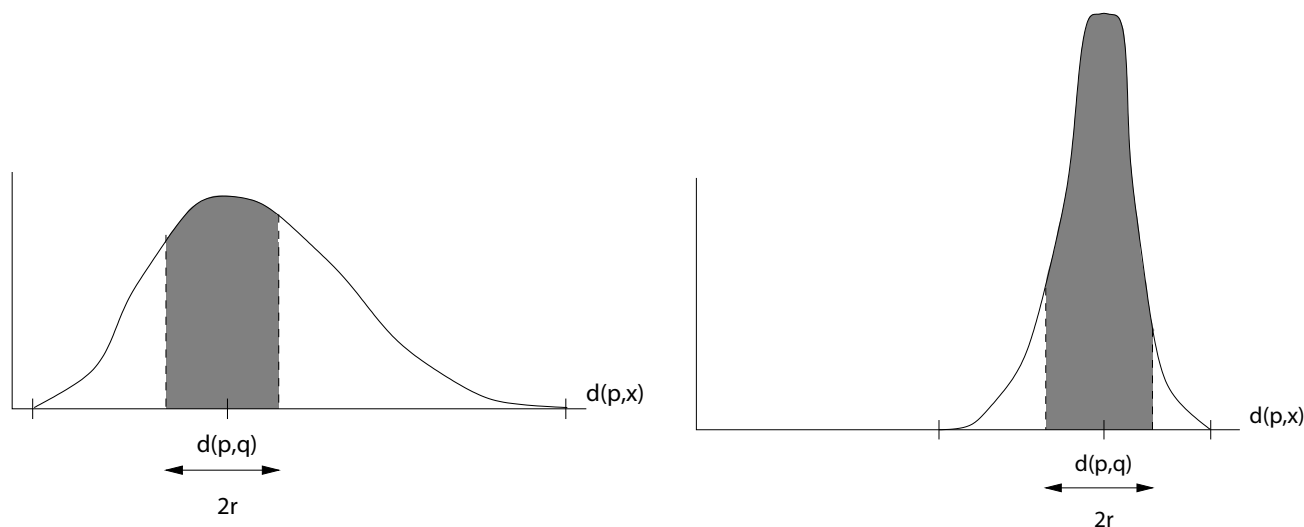
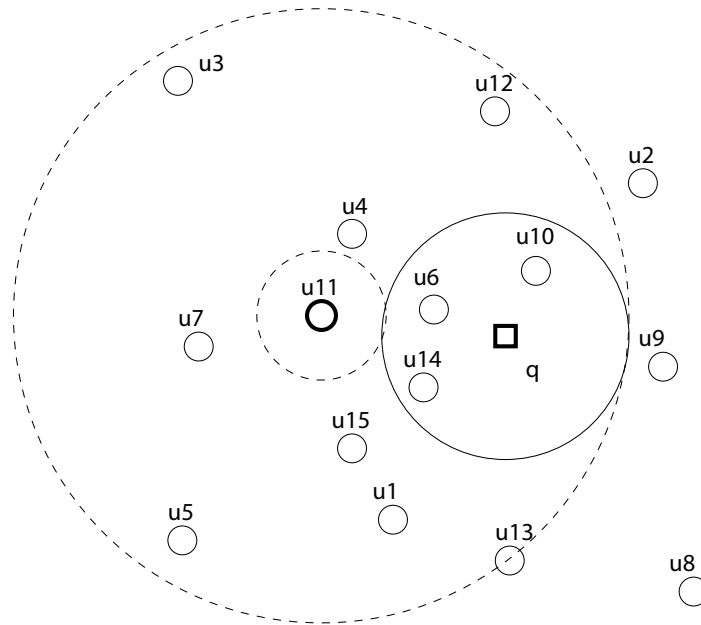


Figure 2. Using pivots to discard elements. Given query  $q$ , with the radius indicated by the solid circle, we should output  $u_6$ ,  $u_{10}$  and  $u_{14}$ . We measure  $d(q, u_{11})$ . Knowing the distances from  $u_{11}$  to the other elements, we are able of discarding  $u_2$ ,  $u_8$  and  $u_9$  without comparing them against  $q$



this intrinsic difficulty (Chávez, Navarro, Baeza-Yates, & Marroquin, 2001).

Using a well-known example, consider a distance such that  $d(x, x) = 0$  and  $d(x, y) = 1$  for all  $x \neq y$ . Using this distance (in fact an equality test), not information from a comparison is obtained, except that the element considered is or is not our query. It is not possible to avoid a sequential search in this case, no matter how smart the algorithm.

Consider the *histogram* of distances between points in the metric space  $X$ . This can be approximated by using the dictionary  $U$  as a random sample of  $X$ . The idea is that, as the space has higher intrinsic dimension, the mean  $m\mu$  of the histogram grows, its variance  $\sigma^2$  is reduced (at least this is the case on random vector spaces), or both. The previous example is an extreme case.

Figure 1 shows an intuitive explanation of why the search problem is harder when the histogram is concentrated. Considering a random query  $q$  and its comparison against some database element  $p$ , the distance  $d(p, q)$  is distributed according to the histogram of the figure. The triangular inequality shows that one can discard any point  $u \in U$  such that  $d(p, u) \notin [d(p, q) - r, d(p, q) + r]$ . The grayed areas in the figure show the points that cannot be discarded. As the histogram is more and more concentrated around its mean, fewer points can be discarded using the information given by  $d(p, q)$ . Moreover, in many

cases (e.g., to retrieve a fixed fraction of the database) the search radius  $r$  must grow as the mean distance  $m\mu$  grows, which makes the problem even harder.

This phenomenon is independent of the nature of the metric space (vectorial or not, in particular) and is a way to quantify how hard it is to search an arbitrary metric space. A very rough but useful definition of intrinsic dimension is  $\rho = \mu^2 / (2\sigma^2)$ . It grows as the median grows or the variance shrinks, as desired. Moreover, when applied to uniformly distributed vector spaces, it matches the classic concept of representational dimension very well, within a constant factor between 1.0 and 1.5 that depends on the type of distance used. Finally, measuring this intrinsic dimension on an arbitrary and unknown metric space can be accomplished by simple statistical means via a reasonable number of distance evaluations among random points of the set. This is much simpler and cheaper than other approaches.

On the other hand, the measure  $\rho$  approximates only roughly the real difficulty of searching a metric space. Several other measures have been proposed with variable success and easiness of use, and the issue is not solved at all. Moreover, the most relevant open issue in metric database searching is how to cope with spaces of high dimension.

## CURRENT ALGORITHMS FOR METRIC DATABASES

Different indexing schemes have been proposed to speed up similarity searches (Chávez, Navarro, Baeza-Yates, & Marroquin, 2001). The idea is to use the triangle inequality to filter out database elements  $u$  that can be proved to be far enough from  $q$  without the need to compute  $d(q,u)$ .

### Pivoting Schemes

A *pivot* is a distinguished database element whose distance to some other elements has been precomputed and stored in an index. Imagine that we have precomputed  $d(p,u)$  for some pivot  $p$  and every  $u \in U$ . At search time, we compute  $d(p,q)$ . Then, by the triangle inequality,  $d(q,u) \geq |d(p,q) - d(p,u)|$ , so if  $|d(p,q) - d(p,u)| > r$  we know that  $d(q,u) > r$ , hence  $u$  will be filtered out without need of computing that distance (see Figure 2).

The most basic scheme chooses  $k$  pivots  $p_1 \dots p_k$  and computes all the distances  $d(p_i, u)$ ,  $u \in U$ , into a table of  $kn$  entries. Then, at query time, all the  $k$  distances  $d(p_i, q)$  are computed and every element  $u$  such that

$$D(q,u) = \max_{i=1,\dots,k} |d(p_i, q) - d(p_i, u)| > r \quad \text{is discarded.}$$

Finally,  $q$  is compared against the elements not discarded.

As  $k$  grows, there are more comparisons against pivots, but  $D(q,u)$  becomes closer to  $d(q,u)$  and more elements may be discarded. It can be shown that there is an optimum number of pivots  $k^*$ , which grows fast with the dimension and becomes quickly unreachable because of memory limitations. In all but the easiest metric spaces, one simply uses as many pivots as memory permits. There exist many variations over the basic idea, including different ways to sort the table of  $kn$  entries to reduce extra CPU time (e.g., see Chávez, Marroquin, & Navarro, 2001).

Although they look completely different, several tree data structures are built on the same pivoting concept (e.g., see Yianilos, 1993). In most of them, a pivot  $p$  is chosen as the tree root, and its subtrees correspond to ranges of  $d(p,u)$  values, being recursively built on the elements they have. In some cases, the exact distances  $d(p,u)$  are not stored, but the range can be inferred from the subtree in which the element  $u$  is found. Albeit this reduces the accuracy of the index, the tree usually takes  $O(n)$  space instead of the  $O(kn)$  needed with  $k$  pivots. Moreover, every internal node is a partial pivot (which knows only distances to its subtree elements), so we actually have many more pivots (albeit local and with coarse data). Finally, the trees can be traversed with sublinear extra CPU time.

Different tree variants arise according to the tree arities, the way the ranges of distances are chosen (i.e., if they are trying to balance the tree or not), how local are the pivots (different nodes can share pivots, which no longer belong to the subtree), the number of pivots per node, and so on. Very little is known about which variant is best. For example, the golden rule of preferring balanced trees, which works well for exact searching, becomes a poorer choice as the dimension increases and the tree becomes unbalanced. For very high-dimensional data, the best data structure is a linked list (i.e., a degenerate tree; Chávez & Navarro, 2000). Also, little is known about how to choose the pivots.

### Local Partitioning Schemes

Another scheme builds on the idea of dividing the database into spatially compact groups, meaning that the elements in each group are close to each other. A representative is chosen from each group, so that comparing  $q$  against the representative has good chances of discarding the whole group without further comparisons. Usually these schemes are hierarchical, and groups are recursively divided into subgroups.

There are two main ways to define the groups. One defines “centers” with a covering radius so that all elements in its group are within the covering radius distance to the center (e.g., see Ciaccia, Patella, & Zezula, 1997). If a group has center  $c$  and covering radius  $r_c$ , then if  $d(q,c) > r + r_c$ , it can be wholly discarded. The geometric shape of this scheme corresponds to a ball around  $c$ . In high dimensions, all the balls tend to overlap, and even a query with radius zero has to enter many balls.

This overlap problem can be mostly alleviated with the second approach (e.g., see Navarro, 2002). The set of centers is chosen and every other point is added to the group of its closest center. At query time, if  $q$  is closest to center  $c_i$ , and  $d(q,c_j) - r > d(q,c_i) + r$ , then we can discard the group of  $c_j$ . The geometric shape in this approach corresponds to a Dirichlet domain of the space (a generalization of the Voronoi diagram to metric spaces), without overlaps among groups.

The fact that balls can overlap has a good counterpart, however. A new point can be inserted inside any ball, enlarging the covering radius if necessary. With the Dirichlet domain, on the other hand, there is exactly one group where a new point must appear.

The flexibility at insertion is important when considering dynamism and secondary memory because a flexible scheme better permits filling disk pages, balancing the tree if desired and reshaping the tree at lower cost. In general, dynamism and secondary memory con-

siderations have become a focus of attention only recently, and only few data structures consider them (e.g., see Ciaccia, Patella, & Zezula, 1997). This is important when considering these structures for serious, massive data management.

Finally, another important trend is that of probabilistic and approximate algorithms (e.g., see Chávez & Navarro, 2003). These algorithms are especially welcome in this area because similarity searching usually means that there are no right and wrong answers, just better and worse ones. Hence, it is acceptable to return suboptimal answers if it is done much faster. In particular, it has been shown that this type of technique is a very powerful tool to cope with the dimensionality curse.

## REFERENCES

- Böhm, C., Berchtold, S., & Keim, D. (2001). Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, 33(3), 322-373.
- Chávez, E., Marroquin, J. L., & Navarro, G. (2001). Fixed queries array: A fast and economical data structure for proximity searching. *Multimedia Tools and Applications (MTAP)*, 14(2), 113-135.
- Chávez, E., Navarro, G., Baeza-Yates, R., & Marroquin J. L. (2003). Proximity searching in metric spaces. *ACM Computing Surveys*, 33(3), 273-321.
- Chávez, E., & Navarro, G. (2000). An effective clustering algorithm to index high dimensional metric spaces. In *Proceedings of the 7th International Symposium on String Processing and Information Retrieval (SPIRE 2000)*, pp. 75-86. IEEE CS Press.
- Chávez, E., & Navarro, G. (2003). Probabilistic proximity search: Fighting the curse of dimensionality in metric spaces. *Information Processing Letters*, 85, 39-46.
- Ciaccia, P., Patella, M., & Zezula, P. (1997). M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd Conference on Very Large Databases (VLDB'97)*, pp. 426-435. Software available for download from <http://www-db.deis.unibo.it/Mtree/>
- Gaede, V., & Günther, O. (1998). Multidimensional access methods. *ACM Computing Surveys*, 30(2), 170-231.
- Navarro, G. (2002). Searching in metric spaces by spatial approximation. *The Very Large Databases Journal*, 11(1), 28-46.
- Peter, N., & Yianilos, P. N. (1993). Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*

## KEY TERMS

**Curse of Dimensionality:** The sharp dependency on the space dimension experimented by any search algorithm on vector or metric spaces.

**Distance:** A function from pairs of objects into non-negative real numbers. It can be zero only if the two arguments are the same. It must be symmetric and obey the triangle inequality.

**Histogram:** Approximation, obtained by sampling, of a distribution (e.g., a distance distribution).

**Index:** A data structure built on a database to speed up searches.

**Intrinsic Difficulty:** A numerical measure of how hard searching a given database is, independently of the index used. It grows with the intrinsic dimension.

**Intrinsic Dimension:** The minimum representational dimension in which the database could be represented without distorting its distances.

**Metric Database:** A database that provides access to a set of objects based on similarity to a query object, using the metric space model.

**Metric Space:** A set of elements together with a distance function defined among pairs of elements.

**Multimedia Data:** Data that represent continuous objects of the real world (e.g., images, audio), in which searching by exact equality is usually meaningless.

**Metric Range Query:** A similarity query that retrieves all elements within some distance to a query object.

**Nearest Neighbor Query:** A similarity query that retrieves the closest elements to some query object.

**Representational Dimension:** The number of coordinates a vector space has.

**Triangle Inequality:** The property  $d(x,y) \leq d(x,z) + d(z,y)$  that a distance  $d$  must satisfy in order to be a metric.



# Modeling and Querying Temporal Data

Abdullah Uz Tansel<sup>1</sup>

Bilkent University, Turkey

## INTRODUCTION

Databases in general store current data. However, the capability to maintain temporal data is a crucial requirement for many organizations and provides the base for organizational intelligence. A *temporal database* has a time dimension and maintains time-varying data (i.e., past, present, and future data). In this article, we focus on the relational data model and address the subtle issues in modeling temporal data, such as comparing database states at two different time points, capturing the periods for concurrent events, and accessing to times beyond these periods, handling multivalued attributes, coalescing, and restructuring temporal data (Gadia 1988, Tansel & Tin, 1997). Many extensions to the relational data model have been proposed for handling temporal data.

There is a growing interest in temporal databases in many application domains. The first book dedicated to temporal databases, by Tansel et al. (1993), was followed by others addressing issues in handling time-varying data (Betini, Jajodia, & Wang, 1988; Date, Darwen, & Lorentzos, 2002; Snodgrass 1999).

## BACKGROUND

The set  $T$  denotes time values, and it is a total order under the " $\leq$ " relationship, hence allowing comparisons and calculations. The set  $T$  can be represented by integers ( $I$ ) or real numbers ( $R$ ). Time is continuous, and real numbers are a better approximation because they can easily accommodate time granularity. Because of its simplicity, time values in a calendar system are commonly implemented by the integers 0, 1, ... *now*. The symbol 0 is the relative origin of time, and *now* is a special symbol that represents the current time. *Now* advances according to the time granularity used. There are different time granularities, such as seconds, minutes, hours, days, months, years, and so forth (for a formal definition, see Betini, Jajodia, & Wang 1988).

A subset of  $T$  is called a *temporal set*. A temporal set that contains consecutive time points  $\{t_1, t_2, \dots, t_n\}$  is represented either as a closed interval  $[t_1, t_n]$  or as a half open interval  $[t_1, t_{n+1})$ . A *temporal element* (Gadia, 1988) is a temporal set that is represented by the maximal intervals corresponding to its subsets having consecu-

tive time points. Temporal sets, intervals, and temporal elements can be used as time stamps for modeling temporal data and are essential constructs in temporal query languages. Temporal sets and temporal elements are closed under set-theoretic operations, whereas intervals are not. However, intervals are easier to implement. Time intervals, hence temporal elements and temporal sets, can be compared. The possible predicates are *before*, *after*, *meet*, *during*, and so forth. An interval or a temporal set (element) that includes *now* expends in its duration. Other symbols, such as *forever* or *until changed*, are also proposed as alternatives to the symbol *now* for intuitive handling of future data.

There are various aspects of time in databases (Snodgrass, 1987). *Valid time* indicates when a data value becomes effective. It is also known as *logical* or *intrinsic time*. On the other hand, the *transaction time* (or *physical time*) indicates when a value is recorded in the database. *User-defined time* is application-specific and is an attribute whose domain is time. Temporal databases are in general append-only that is, new data values are added to the database instead of replacing the old values. A database that supports valid time keeps historical values and is called a *valid time (historical)* database. A *rollback* database supports transaction time and can roll the database back to any time in the past. Valid time and transaction time are orthogonal. However, a temporal database that supports both valid time and transaction time is capable of handling retroactive and post-active changes on temporal data. In the literature, the term *temporal database* is generically used to mean a database with some kind of time support.

This chapter will focus on the valid time aspect of temporal data in relational databases. However, the discussion herein can easily also be extended to databases that support transaction time.

## MODELING TEMPORAL DATA

A *temporal atom* is a time stamped value,  $\langle t, v \rangle$ , and represents a temporal value. It asserts that the value  $v$  is valid over the period of time stamp  $t$  that can be a time point, interval, temporal set, or temporal element. Time points are suitable only for values that are valid at a time point, not over a period. Time can be added to tuples or

Figure 1. Salary in tuple time stamping

<u>E#</u>	<u>SAL</u>	<u>FROM</u>	<u>TO</u>	<u>E#</u>	<u>SAL</u>	<u>TIME</u>
E1	20K	1/01	5/02	E1	20K	{(1/01, 5/02) ∪
E1	20K	8/02	6/03			[8/02, 6/03]}
E1	30K	6/03	now	E1	30K	{(6/03, now]}
<b>a. Intervals</b>				<b>b. Temporal elements</b>		

attributes and hence, temporal atoms can be incorporated differently into the relational data model. To represent temporal atoms in tuple time stamping, a relation is augmented with two attributes that represent the end points of an interval or a time column whose domain is intervals, temporal sets, or temporal elements. Figure 1 depicts salary (SAL) history of an employee (E1), in which intervals or temporal elements are used as time stamps with a time granularity of month/year. Salary is 20K from 1/01 to 5/02 and from 8/02 to 6/03. The discontinuity is the result of the employee quitting on 6/02 and returning on 8/02. The salary is 30K since 6/03. Figure 2 gives the same salary history in attribute time stamping. An attribute value is a set of temporal atoms. Each relation has only one tuple that carries the entire history. It is also possible to create a separate tuple for each time stamped value (temporal atom) in the history (i.e., three tuples for Figure 2.a, two tuples for Figure 2.b).

One noteworthy aspect of data presented in Figure 2 is that the time stamps are glued to attribute values. In other words, attribute values are temporal atoms. In forming new relations as a result of query expressions, these time stamps stay with the attribute values. On the other hand, in tuple time stamping, a time stamp may be implicit (glued) or explicit (unglued) to tuples. This is a design choice, and the relations in Figure 1 can be interpreted as having implicit or explicit time stamps. An implicit time stamp is not available to the user as a column of a relation, though the user can refer to it. On the other hand, an explicit time stamp is like any other attribute of a relation and it is defined on a time domain. Implicit time stamps restricts the time of a new tuple created from two constituent tuples, since each tuple may not keep its own time stamp and a new time stamp needs to be assigned to the resulting tuple. Explicit time stamps allow multiple time stamps in a tuple. In this case, two tuples may be combined to form a new tuple, each carrying its own time reference. However, the user needs to keep track of these separate time references.

## TEMPORAL RELATIONS

Figure 3 shows some sample employee data for the EMP relation over the scheme E# (Employee number), ENAME (Employee name), DNAME (Department name) and SALARY. E# and ENAME are (possibly) constant

Figure 2. Salary in attribute time stamping

<u>E#</u>	<u>SAL</u>	<u>E#</u>	<u>SAL</u>
E1	{<(1/01, 5/02), 20K>	E1	{<{(1/01, 5/02) ∪
	<[8/02, 6/03], 20K>		[8/02, 6/03]}, 20K>
	<[6/03, now], 30K>}		<{(6/03, now]}, 30K>}
<b>a. Intervals</b>		<b>b. Temporal elements</b>	

attributes, whereas DNAME and SALARY change over time. In EMP relation, temporal elements are used in temporal atoms for representing temporal data. Time stamp of E# represents the life span of an employee that is stored in the database. Note that EMP is a nested (NINF—Non-First Normal Form) relation. It is one of the many possible relational representations of the employee data (Clifford & Tansel, 1985; Gadia, 1988; Tansel, 2004). Figure 4 gives, in tuple time stamping, three 1NF relations, EMP\_N, EMP\_D, and EMP\_S for the EMP relation of Figure 3 (Lorentzos & Johnson, 1987; Navathe & Ahmed 1987; Sarda, 1987; Snodgrass 1987). In Figure 3, temporal sets (elements) can also be used as the time reference. Similarly, in the relations of Figure 4, intervals, or temporal sets (elements), can also be used as the time reference in a time attribute that replaces the Start and End columns.

Note that in tuple time stamping, a relation may contain only attributes whose values change at the same time; attributes changing at different times require separate relations. Each particular time stamping method imposes restrictions on the type of base relations allowed as well as the new relations that can be generated from the base relations. The EMP relation in Figure 3 is a unique representation of the employee data, in which each tuple contains the entire history of an employee (Clifford & Tansel, 1985; Gadia, 1988; Tansel, 1997). The E# is a temporal grouping identifier, regardless of the time stamp used (Clifford, Croker, & Tuzhilin, 1993). In the case of tuple time stamping an employee's data is dispersed into several tuples (i.e., there are three salary tuples for employee 121 in Figure 4.c). These tuples belong to the same employee because their E# values are equal.

For the relations in Figures 3 and 4 there are many other possible representations that can be obtained by taking subsets of temporal elements (intervals) and creating several tuples for the same employee. These relations are called *weak relations* (Gadia, 1988). Though they contain the same data as the original relation in unique representation, query specification becomes very complex. Weak relations naturally occur in querying a temporal database. Weak relations can be converted to an equivalent unique relation by coalescing tuples that belong to the same object (employee) into one single tuple (Bohlen, Snodgrass & Soo 1996; Sarda, 1987).

Design of relational databases is based on functional and multivalued dependencies. Roughly, a relation is created to represent the data for similar objects (entities),

Figure 3. The EMP relation in attribute time stamping

E#	ENAME	DNAME	SALARY
<[1/01,now], 121>	Tom	<[1/01, 2/02), Sales> <[4/02, 8/02), Mktg>	<[1/01, 5/02), 20K> <[5/02, 7/02), 25K> <[7/02, now], 30K>
<[3/03,8/03), 133>	Ann	<[3/03, 8/03), Sales>	<[3/03, 8/03), 35K>
<[8/02, now], 147>	John	<[8/02, now], Toys>	<[8/02, now], 42K>

Figure 4. The EMP relation in tuple time stamping

E#	NAME
121	Tom
133	Ann
147	John

(a) EMP\_N Relation

E#	DNAME	START	END
121	Sales	1/01	2/02
121	Marketing	4/02	8/02
133	Sales	3/03	8/03
147	Toys	8/03	Now

(b) EMP\_D Relation

E#	SALARY	START	END
121	20K	1/01	5/02
121	25K	5/02	7/02
121	30K	7/02	Now
133	35K	3/03	8/03
147	42K	1/01	Now

(c) EMP\_S Relation

such as employees. Moreover, a separate relation is defined for the many-to-many relationships between entities. Any attribute involved in a multivalued dependency is also placed into a separate relation. In temporal databases, functional dependencies are transformed into multivalued dependencies. Therefore, each time a dependent attribute is placed into a separate relation in the case of tuple time stamping as is seen in Figure 4. If attribute time stamping and nested relations are used all of the time dependent attributes that belong to similar entities, such as employees, can be placed in one relation, as seen in Figure 3. Details of designing nested relations based on functional and multivalued dependencies are discussed in Ozsoyoglu and Yuan (1987) and Tansel and Garnett (1989).

### CRITICAL ISSUES IN MODELING TEMPORAL DATA

Following are the critical issues in modeling temporal data (Tansel & Tin, 1997). Let  $D_t$  denote the database state at time  $t$ :

1. The data model should be capable of modeling and querying the database at any instance of time (i.e.,  $D_t$ ). Note that when  $t$  is *now*,  $D_t$  corresponds to a traditional database.
2. The data model should be capable of modeling and querying the database at two different time points, intervals, and temporal set (elements) (i.e.,  $D_t$  and  $D_{t'}$  where  $t \neq t'$ ).
3. The data model should allow different periods of existence in attributes within a tuple (i.e., nonhomogenous [heterogeneous] tuples should be allowed). In a homogenous temporal relation, all the

4. The data model should allow multivalued attributes at any time point (i.e., in  $D_t$ ).
5. A temporal query language should have the capability to return the same type of objects on which it operates. This may require coalescing several tuples to obtain the desired result.
6. A temporal query language should have the capability to regroup the temporal data according to a different grouping identifier that could be one of the attributes in a temporal relation.
7. The model should be capable of expressing set-theoretic operations, as well as set comparison tests, on the time stamps, be they time points, intervals, or temporal sets (elements).

Issues 3, 5, 6, and 7 need more elaboration, whereas the rest do not require any further justification. Homogeneity (Gadia, 1988) requires that the attribute values of a tuple should be defined over the same period of time. Implicit time attributes, by definition, require homogenous tuples. Let  $r$  be a temporal relation and  $\tau$  and  $\tau'$  be two of its tuples. Let  $\tau_T$  and  $\tau'_T$  be the temporal elements (sets) over which  $\tau$  and  $\tau'$  are defined, respectively. Cartesian product of these two tuples can only be defined over  $\tau_T \cap \tau'_T$ . In other words, from these two tuples, data can be extracted only in the common period defined by  $\tau_T \cap \tau'_T$ . Parts of  $\tau_T$  and  $\tau'_T$  outside of their intersection are not accessible (i.e.,  $\tau_T - \tau'_T$ , or  $\tau'_T - \tau_T$ ). One can set the semantics of a query language to allow a virtual Cartesian product of tuples with different times. Though this allows the interpretation of one single expression, it is not possible to carry the intermediate results from one expression to another.

Ideally, a temporal query language should retrieve relations in unique representation. In case the query language retrieves weak relations, it should have the capability to transform them into equivalent unique relations in unique representations. Consider the scheme of the EMP relation given in Figure 3. Let  $r_1$  be the instance of EMP relation containing Tom's data in the interval [3/01, 10/01), and let  $r_2$  be the relation instance having Tom's data in the interval [6/01, 1/02). Hence,  $r_1 \cup r_2$  contains two tuples that can be coalesced into a single tuple for the time period [3/01, 1/102). The former would be a weak relation, whereas the latter would be a unique relation.

The capability to restructure temporal relations should be provided in a temporal query language. In Figure 3, the employee data is grouped with respect to E#. The employee data of Figure 3 can also be grouped with respect to the department values. This will facilitate answering queries that involve the department attribute.

A sample query might be, “give the E#, name, and salary of the employee for each department” or “does the validity period of any department include [3/01, now]”?

For issue 7, any data model using temporal sets (elements) as time stamps naturally supports set-theoretic operations and set-comparison tests on time stamps. However, the case of time points and intervals is not straightforward; any data model using them should be able to simulate these operations.

## FUTURE TRENDS

Increased hardware and software capacity is paving the way for implementing temporal databases in many application domains. Commercial database vendors will continue adding functionality for temporal support to their software products. Data warehousing capabilities such as maintaining aggregate data are also expected to be added to temporal databases because they contain the source data for these aggregates. However, efficient solution methodologies are needed for the maintenance of the aggregated data.

There are many open issues, especially in the implementation of temporal databases with functionality addressing the issues discussed earlier, such as storage structures, indexing temporal data, and efficient processing of temporal queries.

## CONCLUSION

Modeling of temporal data presented in the previous sections also has implications for the temporal query languages that can be used. There are many language proposals for temporal databases (Lorentzos & Mitsopoulos, 1997; Snodgrass, 1987 1995; Tansel, Arkun, & Ozsoyoglu, 1989). In addition to traditional algebraic operations and calculus constructs, a temporal query language should have projection and selection operations on the time dimension. Moreover, in the case of attribute time stamping, restructuring operations, such as *nest* and *unnest*, are also needed (Tansel, 1997). Operations to form new temporal atoms, depending on how they are incorporated into attribute and tuple time stamping, are also needed. SQL2 (Structured Query Language) has a time domain capability for implementing tuple time stamping with intervals. SQL3 has the capabilities for implementing temporal elements as well as for attributing time stamping. However, what temporal constructs should be added to SQL3 is an open issue.

## ACKNOWLEDGMENT

Research is supported by the grant# 65406-00-34 from the PSC-CUNY Research Award Program.

## REFERENCES

- Betini, C., Jajodia, S., & Wang, S. (1988). *Time granularities in databases, data mining and temporal reasoning*. New York: Springer-Verlag.
- Bohlen, M. H., Snodgrass, R. T., & Soo, M. D. (1996). Coalescing in temporal databases. *Proceedings of the International Conference on Very Large Databases*, Bombay, India (pp. 180-191).
- Clifford, J., Croker, A., & Tuzhilin A. (1993). On completeness of historical data models. *ACM Transactions on Database Systems*, 19(1), 64-116.
- Clifford, J., & Tansel, A. U. (1985). On an algebra for historical relational databases: Two views. *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 247-265).
- Date, C. D., Darwen, H., & Lorentzos, N. (2003). *Temporal data and the relational data model*. San Francisco: Morgan Kaufmann.
- Etzion, O., Jajodia, S., & Sripada, S. (Eds.). (1998). *Temporal databases: Research and practice*. New York: Springer-Verlag.
- Gadia, S. K. (1988). A homogeneous relational model and query languages for temporal databases. *ACM Transactions on Database Systems*, 13(4), 418-448.
- Lorentzos, N. A., & Johnson, R. G. (1988). Extending relational algebra to manipulate temporal data. *Information Systems*, 13(3), 289-296.
- Lorentzos, N. A., & Mitsopoulos, Y. G. (1997). SQL extension for interval data. *IEEE Transactions on Knowledge and Data Engineering*, 9(3), 480-499.
- Navathe, S. B., & Ahmed, R. (1987). TSQL-A language interface for history databases. *Proceedings of the Conference on Temporal Aspects in Information Systems* (pp. 113-128).
- Ozsoyoglu, M. Z., & Yuan, L. -Y. (1987). A new normal form for nested relations. *ACM Transactions on Database Systems*, 12(1), 111-136.



Sarda, N. L. (1987). Extensions to SQL for historical databases. *IEEE Transactions on Software Engineering*, 12(2), 247-298.

Snodgrass, R. T. (1987). The temporal query language Tquel. *ACM Transactions on Database Systems*, 12(2), 247-298.

Snodgrass, R. T. (1995). *The TSQL2 temporal query language*. Norwell, MA: Kluwer Academic.

Snodgrass, R. T. (1999). *Developing time oriented applications in SQL*. San Francisco: Morgan Kaufmann.

Tansel, A. U. (1986). Adding time dimension to relational model and extending relational algebra. *Information Systems*, 11(4), 343-355.

Tansel, A. U. (1997). Temporal relational data model. *IEEE Transactions on Knowledge and Database Engineering*, 9(3), 464-479.

Tansel, A. U. (2004). On handling time-varying data in the relational databases. *Journal of Information and Software Technology*, 46(2), 119-126.

Tansel, A. U., Arkun, M. E., & Ozsoyoglu, G. (1989). Time-by-example query language for historical databases. *IEEE Transactions on Software Engineering*, 15(4), 464-478.

Tansel, A. U., Clifford, J., Gadia, S. H., Jajodia, S., Segev, A., & Snodgrass, R. T. (Eds.). (1993). *Temporal databases: Theory, design and implementation*. Redwood City, CA: Benjamin Cummings.

Tansel, A. U., & Garnett, L. (1989). Nested temporal relations. *Proceedings of ACM SIGMOD International Conference on Management of Data* (pp. 284-293).

Tansel, A. U., & Tin, E. (1997). Expressive power of temporal relational query languages. *IEEE Transactions on Knowledge and Data Engineering*, 9(1), 120-134.

## KEY TERMS

For a detailed coverage of the terminology see (Tansel et al., 1993, appendix A; and Etzion, Jajodia & Sripada, 1998, pp.367-413).

**Coalescing:** Combining tuples whose times are contiguous or overlapping into one tuple whose time reference includes the time of constituent tuples.

**Homogenous Temporal Relation:** Attribute values in any tuple of a relation are all defined with the same period of time. In a heterogeneous relation, attribute values in a tuple may have different time periods of existence.

**Rollback:** Restoring the database (a temporal relation) to a state that is recorded as of a given time point, interval, or temporal element in a database that supports transaction time.

**Temporal Data Model:** A data model with constructs and operations to capture and manipulate temporal data.

**Temporal Database:** A database that has transaction time support, valid time support, or both. In the literature, it is loosely used to mean a database that has some kind of time support.

**Temporal Element:** Union of maximal time intervals in which no two time intervals overlap or meet.

**Time Granularity:** Unit of time, such as seconds, minutes, hours, days, months, years. Time advances by each clock tick according to the granularity used.

**Time Interval (Period):** The consecutive set of time points between a lower bound ( $l$ ) and an upper bound ( $u$ ) where  $l < u$ . The closed interval  $[l, u]$  includes  $l$  and  $u$  whereas the open interval  $(l, u)$  does not include  $l$  and  $u$ . Half-open intervals,  $[l, u)$  or  $(l, u]$  are analogously defined.

**Transaction Time:** Designates the time when data values are recorded in the database.

**Valid Time:** Designates when data values become valid.

## ENDNOTE

<sup>1</sup> On leave from Baruch College, the City University of New York, USA.



# Moving Objects Databases

M. Andrea Rodríguez-Tastets

University of Concepción and University of Chile, Chile

## INTRODUCTION

Moving objects databases are particular cases of spatio-temporal databases that represent and manage changes related to the movement of objects. Unlike spatio-temporal applications associated with geographic phenomena where the identity of geographic features may change over time, in moving objects databases the objects maintain their identities but change their locations or shapes through time. That is, it is the geometric aspect of an object that changes rather than the object itself. Within this domain, the most suitable applications are those where objects are cars, airplanes, or any object with regular movements.

Traditional DBMSs are not well equipped to handle data about moving objects. One of the reasons is that DBMSs assume that data is constant unless an explicit modification occurs, and this assumption is not adequate for handling continuously changing data such as the locations of moving objects. In traditional DBMSs, it is difficult to specify queries about spatial and temporal information. For example, a query such as “retrieve the cars that will intersect at a particular location in an hour” is not easily expressed with SQL. Finally, location of a moving object is inherently imprecise because the location stored in the database cannot always be the actual location of the object (Wolfson, 2002).

Unlike traditional database applications, moving objects applications involve the following requirements, which are a subset of spatio-temporal applications requirements (Pfoser & Tryfona, 1998):

- the need for representing objects, such as moving cars, with a position in space and an existence in time;
- the need to capture the change of position over time. This change of position may be continuous or discrete;
- the need for representing spatial relations among objects in time; and
- the need to specify spatio-temporal integrity constraints.

## BACKGROUND

Initially, research in databases handled time and space separately. It was only in the 1990s that spatio-temporal databases became an area of active research. The evolution of spatio-temporal databases, therefore, of moving objects databases, involves issues at different levels. For example, at the ontological level, the semantics built into a spatio-temporal database must be in agreement with the ontological concepts related to the space and time of moving objects (Frank, 2003). At the conceptual level, spatio-temporal requirements are expressed in terms that are independent of any particular data model. Conceptual modeling of spatio-temporal databases should consider spatial and temporal aspects associated not only with objects, but also with attributes and relationships (Tryfona, Price & Jensen, 2003). At the level of data models and languages, the typical components for representing moving objects are abstract data types and object classes that incorporate spatial and temporal aspects (Güting et al., 2003).

The basic components of spatio-temporal databases are spatial objects, which are finite sets of points in a space (Pfoser & Tryfona, 1998). From a temporal perspective, properties and relationships are considered facts of objects; therefore, they can be assigned truth-values. There are different types of temporal aspects that have been traditionally discussed:

- *valid time* is the time when a fact is true in a modeled reality;
- *transaction time* is the time when an element in the database, which is not necessarily a fact, is part of the current state in the database; and
- *existence time* of an object refers to the time when the object exists in reality.

In the context of moving objects, the valid time of a given object is the current, past, and future position that has been recorded. The transaction time of a position refers to the current and past positions that were recorded as current in the database. The existence time is associ-

ated only with the existence of an object and not with its position.

Different types of models for time are time points vs. time intervals. A time point is an instant in time, whereas a time interval is defined by a start and end point in time. A spatio-temporal database may store events or states. An event usually occurs at a specific time point, without duration; for example, an event may be that a car crashed. A state, in contrast, has duration and is defined for each time point within an interval; for example, the state of a car that is parked.

## ACHIEVEMENTS OF MOVING OBJECTS DATABASES

This section reviews the development of moving objects databases that concern conceptual modeling, logical models and query languages, and spatio-temporal access methods. This review will be the basis for presenting trends for future moving objects databases.

### Conceptual Modeling

Conceptual modeling aims at providing a direct mapping between the perceived real world and its representation. To fulfill this goal, conceptual models should offer constructs sufficiently powerful to express a model of reality. The current proposals of such constructs include, at a minimum, objects types, relationship types, and attributes. For spatio-temporal databases, these constructs are associated with spatial or temporal concepts (Tryfona et al., 2003).

The traditional strategy for spatio-temporal conceptual modeling has been to extend existing models with constructs that accommodate the requirements of spatial and temporal information. An extension of the Entity-Relation model (ER) to a spatio-temporal ER (STER) (Tryfona & Jensen, 1999; Tryfona et al., 2003) incorporates temporal, spatial, and both spatio-temporal aspects in the specification of constructs. Indeed, the STER allows one to model spatial, temporal, and spatio-temporal cases of entities, attributes, and relationships. STER facilitates the integration of file-based and object-based views of the space (Shekhar et al., 1997), the explicit representation of topological relations between objects, as well as, the representations with multiple granularities.

A different approach to modeling moving objects databases is to consider an extension to UML (Price, Tryfona & Jensen, 2000). The Spatio-Temporal UML supports changes of the instantiated UML elements (i.e., objects, associations, and attributes instances) to provide for associated time periods or spatial extents (Tryfona et al., 2003). The extension incorporates a minimum set of

constructs for spatial, temporal, and thematic data, which can model temporal changes in spatial extents or location, changes in the value of attributes across time or space, and composite data whose components vary depending on time or location. These constructs can then be applied to any UML class diagram and UML model element. The specification of the spatio-temporal semantics of time units (e.g., instants and intervals), time and space dimensions (e.g., existence, valid, and transaction time), models (e.g., object versus field-based space models), and interpolations (e.g., discrete, linear, or spline) are given in specification boxes, which can be associated with any icons or combinations using a unique naming label.

A hybrid ER/OO model for applications with spatio-temporal features, called Modeling of Application Data with Spatio-temporal features (MADS), was explored in (Parent, Spaccapietra & Zimányi, 1999). In this approach, an object-based model is extended with predefined hierarchies of spatial and temporal abstract data types and spatial complex data types to describe the properties of attributes (i.e., name, cardinality, temporal, and spatial dimensions). Spatio-temporal features in MADS can also be associated with objects, attributes, and relationships. The spatial features of MADS supports the discrete or continuous view of space where the spatial domain for any space-varying information is the geometry of any selected item. Relationships in MADS may be of different types, such as is-a relationships, aggregation relationships, constraint relationships (i.e., spatial and synchronization relationships), and dynamic relationships (i.e., transition and generation relationships). This model does not directly support data elements associated with several different spatial extents. In such cases, the data element must be modeled as an association of spatial objects.

### Data Models and Languages

Some practical solutions to the modeling and querying of moving objects propose extensions based on abstract data types (ADTs) (Forlizze et al., 2000; Güting et al., 2000; Güting et al., 2003). They model *moving points* and *moving objects* as three-dimensional (2D + time) or higher dimensional entities whose structure and behavior is captured in an ADT. Designing types and operations to represent moving objects, however, may also require types other than moving points and moving regions (e.g., lines for modeling trajectory). Once the ADTs are defined, they can be integrated into relational or object-oriented databases, and their operations can be used in queries.

The ADT approach focuses on modeling spatial and temporal relationships that can be described with algebraic geometry. Additional operations are also included into ADTs for computing velocity, derivative, turn, and

speed. An extension to ADTs that deals with the modeling of moving objects is to define spatio-temporal predicates and their composition in order to describe the development of relationships between moving objects (Güting et al., 2003). For example, two moving objects may be first 10 miles from each other, then intersect, and finally be 10 miles from each other again.

Constraint databases represent another alternative to data modeling for spatial and temporal data. The idea is to use a mathematical formulation to represent spatial and temporal data as an infinite collection of points that satisfy first-order formulae. The constraint-based model allows a uniform representation of all kinds of spatio-temporal data and supports declarative query languages that are well-suited for complex spatio-temporal queries (Grumbach et al., 2003).

There are different constraint-based models for spatio-temporal data. In the original constraint data model, constraints are expressed as linear equations or inequalities (Kanellakis et al., 1995). The indefinite constraint data model (Koubarakis, 1994) extends the original constraint model to handle indefinite information by defining *possible worlds* as linear or polynomial constraint relations that use variables to handle vague or imprecise values. An extension of the linear constraint model uses differential geometry (Su, Xu & Ibarra, 2001). Within such an approach, simple primitives of velocity and acceleration along with vector operators are sufficient to express relations about movements and the topological and temporal relations among objects. Finally, another extension to the original constraint model treats spatio-temporal data as (possibly infinite) sets of points in a multidimensional space of rational numbers, with no limitation of the space dimension (Grumbach et al., 2003). These sets are then used as values in tuples.

## Access Methods

Traditional access methods do not support spatio-temporal data; therefore, different proposals have been developed to simultaneously support time and space. They aim at indexing objects that move in a two-dimensional space (Mokbel, Ghanem, & Aref, 2003; Di Pasquale et al., 2003). Most of these methods extend spatial access methods to include temporal components. These methods may be classified based on the type of data about moving objects that they deal with. Some methods focus on the retrieval of historical data with queries defined by *timeslices* or *intervals*. Examples are RT-Tree (Xu, Han & Lu, 1990), HR-Tree (Nascimientto et al., 1999), and their variations. A second type of access methods focuses on the trajectory of moving objects, such as TB-Tree and STR-Tree (Pfoser, Jensen & Theodoridis, 2000). Finally, some methods are used to retrieve future positions of moving objects based

on the current positions and movement patterns (Agarwal, Arge & Erickson, 2000, Kollios, Gunopulos & Tsotras, 1999).

A different classification of access methods considers how time is included in the spatial structure. Considering time as another dimension, the 3D-Rtree (Theodoridis, Vazigiannis & Sellis, 1996) treats spatio-temporal data as three-dimensional structures, such that the spatial locations are depicted, say, on the xy-plane, and the z-axis is the time dimension. Another strategy incorporates temporal data in the nodes of the spatial structure, such as the case of the RT-Tree (Xu et al., 1990). Other approaches take into account the overlapping of common data across time, thus, reducing unnecessary duplication of data. This is the type of approach used by HR-Tree (Nascimientto et al., 1999) and OLQ (Tzouramanis, Vassilakopoulos & Manolopoulos, 2000).

In addition to the general spatio-temporal indexing methods, some access methods have been designed taking into consideration specific requirements of the applications where they are used (Fretzos, 2003; Pfoser, Jensen, & Theodoridis, 2000). Such special requirements may include movements within partitions of the space, movements at different speeds, objects as points regardless of their shapes, and dynamic indexing when new objects are included in the system.

## FUTURE TRENDS

Moving objects databases is an important area of database research that has become especially active since the 1990s. Although there have been advances in the different aspects involved in the design of moving objects databases, there are still many challenges that current solutions have either only partially addressed or have not addressed at all. These challenges are summarized in Table 1.

## CONCLUSION

Moving objects databases are a challenging area for further research. Emerging commercial location-based services that make use of devices such as smart cell phones, wireless modems, and GPS devices, whose prices are dropping rapidly, suggest that the use of this technology will spread worldwide. The issues discussed in this article about spatio-temporal concepts, conceptual modeling, data modeling, and access methods emphasize the current state of the start of moving objects databases. Given that moving objects databases are still a young technology, there is a need for further studies that address not only the already-covered aspects of model-

Table 1. Challenges for moving objects applications

<p><b>Uncertainty</b> Moving objects applications should include aspects of uncertainty and quality of data, as well as more semantics through constraints.</p> <p><b>Discrete observations vs. continuous movement</b> Data comes as discrete observations of a continuous movement. This is still a challenging issue for moving objects applications that need to define the frequency of observations.</p> <p><b>Lack of standardization</b> There are still a large number of unstandardized features being used to model spatio-temporal data. In this context, creators of spatio-temporal models should consider how their models may be integrated with ISO TC 211, GML (The Geographic Markup Language)-related developments.</p> <p><b>Data integration</b> Like traditional databases, data integration in moving objects applications presents issues at the semantic, logical, and physical levels of a database.</p> <p><b>Distributed systems</b> Centralized databases may be impractical, in which case, a distributed solution would be needed. Such distributed solution requires a system to allocate, update, and query moving objects or trajectories in distributed data repositories.</p> <p><b>Granularity</b> Not only spatial data, but also temporal data can be viewed at multiple granularities. Such granularities offer multiple</p>	<p>representations of moving objects data, which have an impact on the consistency and integration of databases.</p> <p><b>Consistency</b> Management of consistency is usually accomplished by defining integrity constraints. There are no studies that address deeply enough the specification and modeling of spatio-temporal constraints (rules).</p> <p><b>Query processing</b> Query processing needs to take into account more than indexes. In particular, it also needs to consider optimizers and join algorithms.</p> <p><b>Performance evaluation</b> More research is necessary to verify the performance of existing or new indexing methods in realistic scenarios.</p> <p><b>Aggregation and visualization</b> Spatio-temporal OLAP involves the study of aggregation functions, as well as the visualization of multidimensional data.</p> <p><b>User interface</b> Although there exist several approaches to visual query languages for spatial databases, they all allow querying of only static spatial situations. Querying and visualization of object changes and changes of objects' relationships are the goal for a spatio-temporal language.</p> <p><b>Moving-objects data mining</b>The mining of motion patterns means the detection of periodicity in the movements of objects. Such periodicity may be cyclic (e.g., monthly, diary, and so on) or not, which is useful in the prediction of motion.</p>
---	---

ing and indexing structures, but also new considerations about data mining, warehousing, query processing, consistency, data integration, and user interfaces.

## REFERENCES

Agarwal, P., Arge, L., & Erickson, J. (2000). Indexing moving points. *Symposium on principles of database systems* (pp. 175-186). ACM Press.

Forlizze, L., Güting, H., Nardelli, E., & Schneider, M. (2000). A data model and data structure for moving objects databases (pp. 319-330). *ACM SIGMOD*. ACM Press.

Frank, A. (2003). Ontology for spatio-temporal databases. In K. Koukarakis, R. Sellis, A. Frank, S. Grumbach, R. Güting, C. Jensen, N., et al. (Eds.), *Spatio-temporal databases: The chorochronos approach*, LNCS 2520 (pp. 9-78). Berlin: Springer-Verlag.

Frentzos, E. (2003). Indexing objects moving on fixed networks. In T. Hadzilacos, Y. Manolopoulos, J.

Roddick, & Y. Theodoridis (Eds.), *Advances in spatial and temporal databases*, LNCS 2750 (pp. 289-305). Berlin: Springer-Verlag.

Grumbach, S., Koubarakis, M., Rigaux, P., Scholl, M. & Skiadopoulos, S. (2003). Spatio-temporal models and languages: An approach based on constraints. In K. Koukarakis, R. Sellis, A. Frank, S. Grumbach, R. Güting, C. Jensen, N., et al. (Eds.), *Spatio-temporal databases: The chorochronos approach*, LNCS 2520 (pp. 177-201). Berlin: Springer-Verlag.

Güting, R., Böhlen, M., Erwing, M., Jensen, C., Lorentzos, N., Nardelli, E., et al. (2003). Spatio-temporal models and languages: An approach based on data types. In K. Koukarakis, R. Sellis, A. Frank, S. Grumbach, R. Güting, C. Jensen, N., et al. (Eds.), *Spatio-temporal databases: The chorochronos approach*, LNCS 2520 (pp. 117-176). Berlin: Springer-Verlag.

Güting, R., Böhlen, M., Erwing, M., Jensen, C., Lorentzos, N., Schneider, M., et al. (2000). A foundation for representing and querying moving objects. *ACM Transactions on Database Systems*, 25, 1-42.

Kanellakis, P., Kuper, G., & Revesz, P. (1990). Constraint query languages. *Proceedings of the ACM Symposium on Principles of Database Systems* (pp. 299-313). ACM Press.



- Kollios, G., Gunopulos, D., & Tsotras, V. (1999). On indexing mobile objects. *Proceedings of the ACM Symposium on Principles of Database Systems* (pp. 261-272). ACM Press.
- Koubarakis, M. (1994). Database models for infinite and indefinite temporal information. *Information Systems*, 19, 141-173.
- Mokbel, M., Ghanem, T., & Aref, W. (2003). Spatiotemporal access methods. *IEEE Data Engineering Bulletin*, 26, 40-49.
- Mokbel, M.F., & Aref, W.G. (2003). Spatiotemporal access methods. *IEEE Data Engineering Bulletin*, 26, 40-49.
- Nascimento, M., Jefferson, R., Silva, J., & Theodoridis, Y. (1999). Evaluation of access structures for discretely moving points. *Proceedings of the Workshop on Spatiotemporal Database Management* (pp. 171-188).
- Parent, C., Spaccapietra, S., & Zimányi, E. (1999). Spatiotemporal conceptual models: Data structure + space + time. In C. Medeiros (Ed.), *ACM GIS* (pp.26-33). ACM Press.
- Pfoser, D., Jensen, C., & Theodoridis, T. (2000). Novel approaches in query processing for moving object trajectories, *International Conference on Very Large Data Bases* (pp. 395-406). Morgan Kaufmann Publishers.
- Pfoser, D. & Tryfona, N. (1998). Requirements, definitions and notations for spatiotemporal application environments. *ACM GIS* (pp. 124-130). ACM Press.
- Price, R., Tryfona, N., & Jensen, C. (2000). Extended spatiotemporal UML: Motivations, requirements and constructs. *Journal of Database Management*, 11, 14-27.
- Shekhar, S., Coyle, M., Goyal, B., Liu, D.-R., & Sakar, S. (1997). Data models in geographic information systems. *Communications ACM*, 40, 103-111.
- Su, J., Xu, H., & Ibarra, O. (2001). Moving objects: Logical relationships and queries. In C. Jensen, M. Schneider, & J.T. Vassilis (Eds.), *Advances in spatial and temporal databases*, LNCS 2121 (pp. 3-19). Berlin: Springer-Verlag.
- Theodoridis, Y., Vazirgiannis, M., & Sellis, T. (1996). Spatiotemporal indexing for large multimedia applications. *Proceedings of the International Conference on Multimedia Computing and Systems* (pp. 441-448).
- Tryfona, N., & Jensen, C. (1999). Conceptual data modeling for spatiotemporal applications. *GeoInformatica*, 3, 245-268.
- Tryfona, N., Price, R., & Jensen, C. (2003). Conceptual models for spatio-temporal applications. In K. Koubarakis, R. Sellis, A. Frank, S. Grumbach, R. Güting, C. Jensen, N., et al. (Eds.), *Spatio-temporal databases: The chorochronos approach*, LNCS 2520 (pp. 79-116). Berlin: Springer-Verlag.
- Tzouramanis, T., Vassilakopoulos, M., & Manolopoulos, Y. (2000). Overlapping linear quadtrees and a spatiotemporal query processing. *The Computer Journal*, 43, 325-343.
- Wolfson, O. (2002). Moving objects information management: The database challenge. *Proceedings of the 5th Workshop on Next Generation Information Technology and Systems* (pp. 75-89). Berlin: Springer-Verlag.
- Xu, X., Han J., & Lu, W. (1990). RT-Tree: An improved R-Tree index structure for spatiotemporal database. *Proceedings of the 4th International Symposium on Spatial Data Handling* (pp. 1040-1049).

## KEY TERMS

**Bi-Temporal Query:** Refers to a query that involves both valid and transaction time points or intervals.

**Existence Time:** The time when an object exists in the reality. It can be seen as the valid time of the existence of an object, that is, the valid time of “object *o* exists.”

**Interval-Based Model of Time:** temporality is specified using regular or irregular intervals or periods, which are durative temporal references.

**Point-Based Model of Time:** temporality is specified using explicit occurrences of an event, observation or action, which are punctual occurrences.

**Spatio-Temporal Join Query:** It refers to a query that asks for data that satisfy a spatial relation and a temporal window. For example, find all cars within 10 miles of each other during a given hour (or at 1 pm.).

**Spatio-Temporal Selection Query:** It refers to a query that asks for data that intersect a spatial and temporal window.

**Timeslice or Snapshot Query:** Refers to a query that asks for data as of a given transaction time. For example, find all objects alive during a time interval.

**Transaction Time:** The time when an element (i.e., anything that may be stored in a database) is part of the current state of the database. It can be seen as the valid time of an element that is currently in the database, that is, valid time of “element *e* is current in the database.”

**Valid Time:** The time when a fact (i.e., a statement with an associated truth value) is true in the modeled reality.



# Multilevel Databases<sup>1</sup>

Alban Gabillon

*IUT de Mont de Marsan, France*

## INTRODUCTION

In the context of multilevel security, every piece of information is associated with a *classification* level, and every user is associated with a *clearance* level. The classification and clearance levels are taken from the same set of *security levels*. This set is totally or partially ordered and forms a lattice. The ordering relation is called the *dominance* relation and is denoted by  $\geq$ . An example of a totally ordered set is {Unclassified, Confidential, Secret} with Secret  $>$  Confidential  $>$  Unclassified. An example for a partially ordered set is {low, (Secret, NATO), (Secret, Defence), high} with (Secret, NATO)  $>$  low, (Secret, Defence)  $>$  low, high  $>$  (Secret, NATO), high  $>$  (Secret, Defence), (Secret, NATO) and (Secret, Defence) are incomparable.

In multilevel security, the *security policy* (also called the *confidentiality policy*) states that a user has the permission to *know* a given piece of information only if the clearance level of that user dominates (i.e., is higher than or equal to) the classification level associated with the piece of information. Multilevel security is traditionally opposed to *discretionary security*. In discretionary security, the security rules (i.e., permissions or prohibitions) explicitly refer to users' identities.

## BACKGROUND

Most of the existing multilevel security models for databases are based on the following properties (Bell & LaPadula, 1975):

- **No read up:** This rule states that a subject at level X cannot read information at level Y if Y strictly dominates X.
- **No write down:** This rule states that a subject at level X cannot write information at level Y if X strictly dominates Y.

In these rules, *subject* refers to a user or a process. The process level is the *working level* of the user, on behalf of which that process executes. The level at which the user decides to work can be his or her clearance level or any level that is dominated by the user's clearance level. The no write down restriction is necessary to prevent high

level processes from illegally disclosing sensitive data. Consider a program that contains a Trojan horse, and assume a user with a high clearance decides to work at a high security level and run that program without knowing that it is infected. Without the no write down restriction, the Trojan horse would be capable of writing high classified data into a low level information container, making the high level data available to users with low clearance.

The Bell and LaPadula (1975) rules do not disallow write up. However, in multilevel databases, write up is very often prohibited because of the integrity problems arising from its blind nature (Thomas & Sandhu, 1993). The Bell and LaPadula properties are necessary to enforce the confidentiality policy, but they are not sufficient. Indeed, it is possible to illegally transmit data by other means than simple read and write operations. In the literature, such unauthorized communication paths are referred to as *covert channels*. Covert channels can be of several types: timing channel, inference channel, and signaling channel (National Computer Security Center, 1993).

Multilevel security is for applications requiring a high confidentiality level. It is mainly used by military organizations, and it is sometimes called *military security*. Many multilevel security models have been proposed for multilevel relational databases (see Denning, Lunt, Schell, Shockley, & Heckman, 1988; Haig, O'Brien, Stachour, & Troups, 1990; Jajodia & Sandhu, 1991b; Jukic, Vrbsky, Parrish, Dixon, & Jukic, 1999; Qian & Lunt, 1996; Sandhu & Chen, 1998; Smith & Winslett, 1992) and object-oriented databases (see Jajodia & Kogan, 1990a; Keefe, Tsai, & Thuraisingham, 1989; Lunt, 1990; Millen & Lunt, 1992; Cuppens, Gabillon & Yordanian, 1993, 1997, 1999). Experience has shown that when designing a multilevel security model for a database, the following issues must be addressed:

- Granularity of the data must be defined. This means the data structures that will later be classified must be identified. For example, in the context of relational databases, should security levels be assigned to tables? To rows? To attributes? To attribute values? Once the information granularity is defined, the semantics of the association between a security level and a granule must be specified.

- Inference channels must be prevented. When labeling the data with security levels, one should make sure that no sensitive data can be derived from low level data.
- Multilevel database must be decomposed into a collection of single-level views. Indeed, each user has to be provided with a consistent and complete view of the multilevel database that is compatible with his or her security level.
- Operational semantics for the different update operations must be defined.

Throughout this paper I shall illustrate these points with a small multilevel relational database reduced to a single relation and shall consider the following set of security levels: {Unclassified (U), Confidential (C), Secret (S)}.

## MULTILEVEL DATABASE DESIGN

### Information Granularity

When designing a multilevel security model for a database, the first problem is to define the information granularity. The finer the information granularity, the better *the expressive power* of the model. For example, a model for relational databases in which only tables or rows can be classified would be a model with a poor expressive power, whereas a model in which attribute values (i.e., intersection between a row and a column) can be classified would be a model with a great expressive power.

In fact, defining the information granularity depends on the application needs. To illustrate my point, consider the Wing relation (see Table 1). Primary key of this relation is {Name}.

This relation represents an imaginary French American wing ready for attack. The military security administrator of this database knows the various sensitivities of the data contained in this table:

- The existence of the wing is unclassified.
- The existence of each plane except Firefox is unclassified.
- The existence of Firefox is confidential.

Table 1. Wing

Name	Speed	Range	Objective
F16 Falcon	2145 km/h	2000 km	Target A
Mirage 2000	2340 km/h	1480 km	Target B
Rafale	2124 km/h	1824 km	Target A
X-43Z	8000 km/h	9000 km	Target A
Firefox	12000 km/h	13000 km	Target B

- The existence of the Name, Speed and Range attributes in the Wing table is unclassified.
- The fact that the wing is to launch an attack is confidential (i.e., the existence of the Objective attribute in the Wing table is confidential).
- The speed and the range of each supersonic plane are unclassified
- The speed and the range of X-43Z are confidential.
- The speed of Firefox is secret and the range is confidential.
- Mirage 2000 and Firefox objectives are secret. Other plane objectives are confidential.

Analyzing these facts suggests a model in which tables, attributes, and attribute values can be classified. The security level assigned to a table protects the existence of the table. The security level assigned to an attribute (i.e., column) protects the existence of the attribute in the table. The security level assigned to a primary key value protects the value itself, and therefore the existence of the entity that is identified by the value. The security level assigned to a nonkey attribute value protects the value itself.

Knowing this, data contained in the wing relation are classified, as shown in Table 2.

From this simple example is one important lesson: Designing a multilevel security model with a fine granularity is not an insuperable problem, provided that the semantics of the association between a granule of information and a security level is precisely defined. This approach was not always followed. For example, *Sea View* (Denning et al., 1988), which is one of the first multilevel security models for relational databases, assigns security levels to rows as well as to primary key values. Unfortunately, the semantics of such associations is not clearly defined.

### Inference Control

Again, the Bell and LaPadula (1975) rules are necessary to enforce the confidentiality policy, but they are not sufficient. Covert channels can be used to disclose sensitive data. Many types of covert channels cannot be represented in the security model because many of them are due

Table 2. Wing (U)

Name (U)	Speed (U)	Range (U)	Objective (C)
F16 Falcon (U)	2145 km/h (U)	2000 km (U)	Target A (C)
Mirage 2000 (U)	2340 km/h (U)	1480 km (U)	Target B (S)
Rafale (U)	2124 km/h (U)	1824 km (U)	Target A (C)
X-43Z (U)	8000 km/h (C)	9000 km (C)	Target A (C)
Firefox (C)	12000 km/h (S)	13000 km (C)	Target B (S)

to implementation flaws. However, detecting and eliminating potential inference channels at the model stage is perfectly possible. If high classified data can be reduced from low classified data then there is inference channel. To prevent unauthorized inferences, labelling the data with security levels has to be done carefully. For example, the knowledge of “Objective is an attribute of the Wing table” discloses the knowledge of “the Wing table exists.” Consequently, classifying the Wing table at a level that strictly dominates the level protecting the Objective attribute would create an inference channel. In fact, there should be an *inference control rule* in the security model saying that the security level protecting an attribute has to dominate the security level protecting the table. Another example of potential inference channel is the following: The knowledge that “the speed of Firefox is 6000 km” discloses the existence of Firefox. Consequently, classifying the primary key value Firefox at a level that strictly dominates the level protecting the speed value of Firefox would create an inference channel. Therefore, there should be an inference control rule in the model stating that the security level protecting an attribute value of a given row has to dominate the security level protecting the primary key value of that row.

Several such inference control rules have to be enforced when classifying the data in a relational database. The purpose of this paper is not to discover all these rules. A method to formally derive all the rules as well as one example of application of this method on object-oriented databases is presented in (Cuppens & Gabillon, 1998, 1999). Let us mention that this method would allow us to derive one inference control rule from each integrity constraint.

Of course, not all inference channels can be detected and eliminated. In particular, detecting unauthorized inferences from a knowledge that is not represented in the database is impossible.

### Database Architecture

The Air Force Summer Study (Air Force Studies Board, 1983) suggested three different architectures for building secure multilevel database management systems. These architectures differ from the way the multilevel data are physically stored. The first architecture is called the kernelized database management system (DBMS). In this architecture, the multilevel database is partitioned into a collection of single-level segments. The second architecture is called the distributed (or replicated) DBMS. In this architecture, there is a single-level database at every security level. Each *l*-level database contains the data whose classification level is dominated by *l*. The third architecture is based on the integrity lock technology, but, as mentioned in (Jajodia & Kogan, 1990b), this architecture

is vulnerable to Trojan horses. Hence, this approach cannot be used for highly assured multilevel DBMSs.

Whether the architecture is kernelized or distributed, one needs to decompose the multilevel relation into a collection of single-level relations. Such decomposition is not trivial, and several algorithms have been proposed in the literature (Denning et al., 1988; Jajodia & Sandhu, 1990a, 1991a). In this paper, I propose some simple guidelines that should be followed for decomposing the multilevel database. These guidelines are suggested in the multiview model (Cuppens & Gabillon, 1997, 1999; Cuppens, Gabillon, & Yazdanian, 1993). This model is based on the replicated architecture. The main advantage of this architecture over the kernelized approach is that there is no need to combine the data from several sources when answering queries. In fact, the replicated architecture is the best architecture for a fine-grained multilevel database. The principle of the replicated architecture is the following: Each granule classified at level *l* is stored in the *l*-level database and is replicated in every database that has a level dominating level *l*. Application of this principle to the multilevel Wing relation (see Table 2) gives the following three relations (see Tables 3, 4, and 5).

The unclassified view (see Table 3) of the multilevel Wing relation contains only unclassified data. This view is inconsistent because there are no values for the speed and the range of X-43Z. The confidential view (see Table 4) of the multilevel Wing relation contains unclassified and confidential data. This view is inconsistent because there are also some missing values. The secret view (see Table 5) of the multilevel wing relation contains unclassified, confidential, and secret data and is consistent and complete.

After decomposing the original conceptual multilevel relation, the security administrator (SA) has two solutions to provide unclassified and confidential users with consistent views of the database without disclosing sensitive data:

- The SA thinks that it is acceptable to inform low level users that there are some sensitive values they are not permitted to see. In that case, the SA inserts the special value RESTRICTED each time there is a missing value. Using the RESTRICTED value was first suggested in Sandhu and Jajodia (1992). Semantics of this value is “the value exists

Table 3. Wing (in database at level U)

Name	Speed	Range
F16 Falcon	2145 km/h	2000 km
Mirage 2000	2340 km/h	1480 km
Rafale	2124 km/h	1824 km
X-43Z		

but you are not permitted to know it.” Table 6 shows the unclassified view of the wing relation after the SA has decided to use the RESTRICTED value for the speed and the range of X-43Z.

- Now, in some cases, the SA may judge that knowing the existence of the sensitive value is itself sensitive information. In that case, the SA inserts a *cover story* instead of RESTRICTED. A cover story is a lie that hides the existence of sensitive data. Table 7 shows the confidential view of the Wing relation after the SA has decided to insert a cover story (Target A) each time there was a missing objective value. This means that the SA considers that the existence of a secret objective should not be disclosed. On the contrary, the SA decided to use the RESTRICTED value for the speed of Firefox.

Table 6 is the unclassified view of the multilevel relation represented in Table 2. The users working at the unclassified level express queries on this view. Table 7 is the confidential view on which users working at the confidential level may express queries. However, confidential users also have the possibility to query the unclassified view. Table 5 is the secret view. The users working at the secret level may express queries on this view, but they may also query the confidential and the unclassified views.

Compared to this approach, other existing decomposition algorithms are complex. This is because those algorithms take *polyinstantiated* multilevel relations as input. A multilevel relation is polyinstantiated if it contains different rows with the same key, each at a different classification level. This occurs when the multilevel relation contains some cover stories.

The advantage of the approach described in this paper is that the starting multilevel relation (Table 2) is *not* polyinstantiated. The starting relation reflects precisely

the existing multilevel world. The SA inserts cover stories *after* the decomposition of the starting multilevel relation.

Now, after the insertion of the cover stories, the combination of the three relations (see Tables 5, 6, and 7) represents a polyinstantiated multilevel world that differs from the original multilevel world. In this polyinstantiated world, the Mirage 2000 and the Firefox have one confidential objective and one secret objective. The polyinstantiation technique states that, in case of conflict, the low classified data shall automatically be interpreted as cover stories. Therefore, if a secret user sees the secret database and the confidential database, then he or she shall interpret the confidential objectives of the Mirage 2000 and the Firefox as being lies for confidential users.

Note that the polyinstantiation technique has some drawbacks. It relies on a particular interpretation of the data, and it does not work well in case of a partial order on the set of security levels. A new technique for managing cover stories is presented in Cuppens and Gabillon (2001).

### Updating the Database

Defining the operational semantics for update operations on a multilevel database is not an easy task. It becomes a real challenge if the multilevel relations are polyinstantiated (see Jajodia & Sandhu, 1990b).

The architecture of our database is replicated. The multilevel world is represented by a set of single-level databases. Updating each of these single-level databases can be done via standard SQL operations because each single level database behaves as a nonprotected database. When updating, users interact with the database that corresponds to their working level. The replicated architecture requires that low level updates propagate to the higher levels via a *trusted* replication mechanism.

Table 4. Wing (in database at level C)

Name	Speed	Range	Objective
F16 Falcon	2145 km/h	2000 km	Target A
Mirage 2000	2340 km/h	1480 km	
Rafale	2124 km/h	1824 km	Target A
X-43Z	8000 km/h	9000 km	Target A
Firefox		13000 km	

Table 5. Wing (in database at level S)

Name	Speed	Range	Objective
F16 Falcon	2145 km/h	2000 km	Target A
Mirage 2000	2340 km/h	1480 km	Target B
Rafale	2124 km/h	1824 km	Target A
X-43Z	8000 km/h	9000 km	Target A
Firefox	12000 km/h	13000 km	Target B

Table 6. Wing (in database at level U)

Name	Speed	Range
F16 Falcon	2145 km/h	2000 km
Mirage 2000	2340 km/h	1480 km
Rafale	2124 km/h	1824 km
X-43Z	<b>RESTRICTED</b>	<b>RESTRICTED</b>

Table 7. Wing (in database at level C)

Name	Speed	Range	Objective
F16 Falcon	2145 km/h	2000 km	Target A
Mirage 2000	2340 km/h	1480 km	Target A
Rafale	2124 km/h	1824 km	Target A
X-43Z	8000 km/h	9000 km	Target A
Firefox	<b>RESTRICTED</b>	13000 km	<b>Target A</b>

## Multilevel Databases

However, this is a general principle that may not be appropriate for particular situations, such as the following

- How can one interpret if a low level user has updated a low level cover story? Should the update propagate to the higher levels, or should the new value be considered as a new cover story?
- How can one interpret if a low level user has inserted a low level row with a primary key value that already exists at higher levels? Should the insertion propagate to higher levels, deleting higher level rows with the same primary key?

Regarding these issues, there is no best solution. It depends on the integrity policy. With a solution that considers that sensitive data are of high integrity, low level updates should propagate to higher levels as long as they do not conflict with higher classified data. Consequently, the trusted replication mechanisms shall not propagate to higher levels a low level update performed on a RESTRICTED value or a cover story. Conversely, with a solution that emphasizes the freshness of the information, low level updates shall systematically propagate to higher levels, possibly deleting existing higher level data conflicting with the new data. Of course, there are several intermediate solutions. In this paper I suggest one solution that has the advantage of being the simplest: *Any low level update statement is reproduced “as such” at the higher levels.* Because each single-level database behaves as a nonprotected database, a low level update statement that is reproduced at a higher level will fail only if it violates one or more integrity constraints.

Here is the application of the solution through some sample SQL statements:

**INSERT statement.** Consider a user working at the unclassified level who inserts a new aircraft in the unclassified Wing relation with the following statement:

Table 8. Wing (in database at level U)

...	...	...
F-22 raptor	2000 km/h	3200 km

Table 9. Wing (in database at level C)

...	...	...	...
F-22 raptor	2000 km/h	3200 km	NULL

Table 10. Wing (in database at level S)

...	...	...	...
F-22 raptor	2000 km/h	3200 km	NULL

```
INSERT INTO Wing VALUES ('F-22 raptor,'2000 km/h,'3200 km');
```

Tables 8,9 and 10 show the resulting multilevel database.

The replication mechanism has reproduced the INSERT statement at the confidential and secret levels. Because the INSERT statement did not include any objective value, the confidential and secret objective values are set to NULL. If the user who has performed the insertion has a confidential clearance level or higher, then that user has now to set his or her working level to confidential in order to assign an objective to the F-22 raptor. If the user who has performed the insertion has an unclassified clearance level, then only another confidential or secret user may assign an objective to the F-22 raptor.

Propagation of an INSERT statement to higher levels will fail only if it violates one or more integrity constraints. For example, if an unclassified user issues a statement inserting a “new” aircraft called Firefox in the unclassified Wing relation, then the statement will succeed at the unclassified level but fail at the confidential level because of a primary key violation<sup>2</sup>.

**UPDATE statement.** Consider a user working at the confidential level who assigns an objective to the F-22 raptor with the following statement:

```
UPDATE Wing SET Objective= 'Target A' WHERE name = 'F-22 Raptor';
```

Tables 11 and 12 show the resulting confidential and secret databases.

The replication mechanism has reproduced the UPDATE statement at the secret level.

Note that in our example, the replication mechanism would succeed in reproducing a low level update on a RESTRICTED value or a cover story since such propagation would not violate any integrity constraint.

**DELETE statement.** Consider a user working at the unclassified level who deletes the F-22 raptor with the following statement:

Table 11. Wing (in database at level C)

...	...	...	...
F-22 raptor	2000 km/h	3200 km	Target A

Table 12. Wing (in database at level S)

...	...	...	...
F-22 raptor	2000 km/h	3200 km	Target A



```
DELETE FROM Wing WHERE name = 'F-22 Raptor';
```

Tables 5, 6, and 7 show the final multilevel database. The replication mechanism has successfully reproduced the DELETE statement at the confidential and secret levels.

Let us mention that the operational semantics for update operations we define in this paper achieves completeness that is, every multilevel database can be constructed by some sequence of update operations (Jajodia & Sandhu, 1991b). However, since every update propagates to the higher levels, single-level relations (including RESTRICTED values and cover stories) have to be created in the ascending order of the security levels.

## FUTURE TRENDS

In the recent years, research on multilevel security has been eclipsed by research on role-based security (see Sandhu, Coyne, Feinstein, & Youman, 1996 for an introduction to role-based security). For designing secure commercial database applications, role-based security has proved to be more flexible than multilevel security. Nevertheless, multilevel security is still needed in applications that require a high degree of confidentiality, such as military applications. One should also mention that research on multilevel security has found new application areas, such as digital rights management architectures (e.g., see Popescu, Crispo, & Tanenbaum, 2004).

## CONCLUSION

While presenting multilevel security and multilevel databases, I have sketched a multilevel security model for relational databases. Compared to existing approaches, the model has a better expressive power because tables, attributes, and attribute values may independently be classified. Moreover, the semantics of an association between a security level and a granule of information is precisely defined, allowing one to easily design conceptual multilevel relations which are not polyinstantiated. The decomposition mechanism proposed is straightforward and is strictly based on the replicated architecture. I used the method developed in Cuppens and Gabillon (1999) to eliminate inference channels while classifying the data. Finally, I proposed a simple operational semantics for the traditional SQL update statements.

## REFERENCES

- Air Force Studies Board. (1983). *Multilevel data management security*. Committee on Multilevel Data Management Security, National Research Council.
- Bell, D. E., & LaPadula L. J. (1975). *Secure computer systems: Unified exposition and multics interpretation* (Tech. Rep. ESD-TR-75-306, MTR 2997). Bedford, MA: MITRE.
- Cuppens, F., & Gabillon, A. (1997). A logical approach to model a multilevel object-oriented database. In P. Samarati & R. Sandhu (Eds.), *Tenth IFIP 11.3 Conference on Database Security: Database Security, X: Status and Prospects*. Como, Italy: Chapman & Hall.
- Cuppens, F., & Gabillon, A. (1998). Rules for building multilevel object-oriented databases. *Proceedings of the 5th European Symposium on Research In Computer Security (ESORICS98)*. Lecture Notes in Computer Sciences. Louvain-la-Neuve, Belgium.
- Cuppens, F., & Gabillon, A. (1999). *Logical foundations of multilevel databases. data and knowledge engineering*, 29, 259-291. Elsevier.
- Cuppens, F., & Gabillon, A. (2001). *Cover story management: Data and knowledge engineering* (Vol 37/2, pp. 177-201). Elsevier.
- Cuppens, F., Gabillon A., & Yazdaniyan, K. (1993). Multiview model for multilevel object-oriented databases. *Proceedings of the 9th Annual Computer Security Applications Conference*, Orlando, FL.
- Denning, D. E., Lunt, T. F., Schell, R. R., Shockley, W. R., & Heckman, M. (1988). The sea view security model. *Proceedings of the 1988 IEEE Symposium on Research in Security and Privacy*.
- Haig, J. T., O'Brien, R. C., Stachour, P. D., & Toups, D. L. (1990). The LDV approach to database security. In *Database Security III, Status and Prospect*. North Holland.
- Jajodia, S., & Kogan, B. (1990a). Integrating an object-oriented data model with multilevel security. *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA.
- Jajodia, S., & Kogan, B. (1990b). Transaction processing in multilevel secure database using the replicated architecture. *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy*, Oakland, CA.

## Multilevel Databases

Jajodia, S., & Sandhu, R. (1990a). Polyinstantiation integrity in multilevel relations. *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA.

Jajodia S., & Sandhu, R. (1990b). Update semantics for multilevel relations. *Proceedings of the IEEE Symposium on Research in Security and Privacy*, 103-112.

Jajodia, S., & Sandhu, R. (1991a). A novel decomposition of Multilevel Relations into Single level Relations. *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA.

Jajodia, S., & Sandhu, R. (1991b). Toward a multilevel secure relational data model. *Proceedings of the 1991 ACM SIGMOD International Conference on Management of Data*.

Jukic, N., Vrbsky, S. V., Parrish, A. S., Dixon, B., & Jukic, B. (1999). A belief-consistent multilevel secure relational data model. *Information Systems*, 24(5).

Keefe, T. F., Tsai, W. T., & Thuraisingham, M. B. (1989). SODA: A secure object-oriented database system. *Computer and Security*, 8(6).

Lunt, T. F. (1990). Multilevel security for object-oriented database systems. In D. L. Spooner & C. Landwehr (Eds.), *Database security III: Status and prospects*. North Holland.

Millen, J. K., & Lunt, T. F. (1992). Security for object-oriented database systems. *Proceedings of the 1992 IEEE Symposium on Research in Security and Privacy*.

National Computer Security Center. (1993, November). A guide to understanding covert channel analysis of trusted systems.

Popescu, B., Crispo, B., & Tanenbaum, A. S. (2004). *Support for multi-level security policies in DRM architectures*. New Security Paradigms Workshop, White Point Beach Resort, Nova Scotia.

Qian, X., & Lunt, T. F. (1996). A Mac policy framework for multilevel databases. *IEEE Transactions on Knowledge and Data Engineering*, 8(1).

Sandhu, R., Coyne, E. J., Feinstein, H. L., & Youman, C. E. (1996). Role-based access control models. *IEEE Computer* 29(2), 38-47.

Sandhu, R., & Chen, F. (1998). The MLR data model. *Transactions on Information and System Security*, 1(1), 93-132.

Sandhu, R., & Jajodia, S. (1992). Polyinstantiation for cover stories. *Proceedings of the European Symposium on Research in Computer Security*, Toulouse, France.

Smith, K., & Winslett, M. (1992). Entity modeling in the MLS relational model. *Proceedings of the 18<sup>th</sup> International Conference on Very Large Data Bases*, Vancouver, British Columbia, Canada.

Thomas, R. K., & Sandhu, R. (1993). Concurrency, synchronization, and scheduling to support high-assurance write up in multilevel object-based computing. *Proceedings of the OOPSLA-93 Conference Workshop on Security for Object-Oriented Systems*, Washington, DC.

## KEY TERMS

**Classification Level:** A security level that represents both the confidentiality degree of the information and its category.

**Clearance Level:** A security level that represents both the trust level of the user and his or her need to know.

**Cover Story:** A lie that is used to hide the existence of high classified data.

**Covert Channel:** An unintended communications path that can be used to transfer information in a manner that violates the security policy.

**Inference Channel:** A particular case of a covert channel that exists when high classified data can be deduced from low classified data.

**Information Granularity:** The data structure that can be classified.

**Multilevel Security (MLS):** Security in which the data are associated with classification levels and the users are associated with clearance levels. Accesses to data are granted by comparing classifications and clearances.

## ENDNOTES

<sup>1</sup> This work has been carried out in the framework of the CASC research project (ACI sécurité Informatique 2003-2006).

<sup>2</sup> Of course, whether the propagation of a low level update to higher levels succeeds or fails should not be observable by the low level user who has performed the update.

# Multimedia Databases

**Mariana Hentea**

*Southwestern Oklahoma State University, USA*

## INTRODUCTION

The Internet and technologies such as high-capacity storage devices, broadband telecommunications systems, and multimedia development software systems have accelerated the development of new applications based on the use of multimedia database systems. A multimedia database is a repository of different data objects (text, numeric values, Boolean values, dates, graphical images, video clips, and sound files). Examples of applications based on multimedia databases include news and stock market information on demand; movies on demand; home shopping; medical systems; trademark, patent, and copyright databases; geographic information systems (GIS); weather forecasting; Computer Aided Design (CAD) systems; architectural design; fabric and fashion design; interior design; photographic libraries; art gallery and museum management; law enforcement; criminal investigations; military reconnaissance and surveillance; scientific experiments; and educational systems.

The real-time nature and different kinds, including the size of multimedia data, cause problems for the design, implementation, and management of multimedia databases. Multimedia data requires use of specific technologies to reduce the size of the media data so it could be stored within the database. For example, NASA's Earth Observation System generates a terabyte (1,000 gigabytes) of data a day. This data comprises images recorded from orbiting satellites by video and infrared cameras that are downloaded to earth. The software chosen by the National Library of Australia is the TeraText™ Database System, providing a single access point to over 600,000 images from 18 Australian cultural institutions, including libraries, museums, archives, and galleries representing images related to Australia's cultural heritage from the late 18th century through to the present day. The TeraText™ Database System is known for its scalability, flexibility, speed, and sophistication. The increasing importance of multimedia applications and the introduction of SQL3 in 1999 have favored changes to traditional relational databases. These resulted in the implementation of features to store and manipulate large objects within object-relational databases. One example is Oracle9i, which is an object-relational database management system that offers ca-

pabilities for holding media data. Through the use of media data types, Oracle interMedia software enables the Oracle9i database management system to manage and deliver image, audio, video, and geographical location data in an integrated fashion with other enterprise information.

The creation, storage, and data modeling techniques for multimedia data objects and retrieval of objects are main issues in the development of multimedia databases. Other issues include authenticity, integrity, media metadata, automatic video capture and editing, mobile media applications, scheduling, and quality of multimedia delivery over a network (Bourbakis, 2004; Shu & Yu, 2004). The following section provides a brief overview of the multimedia retrieval methods.

## MULTIMEDIA QUERIES

The representation, structure, and retrieval of objects evolved in time from simple conventional data to more complex multimedia data objects such as video and audio. At the beginning, the retrieval of objects was treated as a single data item using queries based on indexing approaches and identification of the attributes that describe an object. However, this method proved to be inefficient. Retrieval of images from a multimedia database is based on similarities. Modern computer technology can be used to extract images of objects from pictures and videos, but the ability to identify these objects semantically is beyond the capabilities of current research. Because computers cannot add semantic interpretation to images, this must be added manually. Visual queries based on recognizing objects on the basis of their color, shape, or texture and other graphical characteristics are possible. Objects can be indexed using feature values and retrieved based on the similarity to the feature values of other objects. Feature values can be derived for an entire media object or just a specific part of its content. Features that can be measured include shape, color, texture, initial position, and direction and speed of motion. Features can be difficult to quantify and their identification may require complicated and lengthy techniques for extraction. The measurement value for a feature may be a representation of the shape or a histogram representing the color

distribution of the object. Several of the methods for the extraction of feature values require use of advanced algorithms based on artificial intelligence (AI) techniques such as data mining (Shih & Wang, 2004; Thuraisingham, 2001), artificial neural networks and fuzzy logic (Tsai, McGarry, & Tait, 2003), and computer vision (Dunckley, 2003). These techniques are explored to enhance the feature extraction and increase the quality of multimedia retrieval and processing.

If the retrieval of images is to be based on keywords, then either the images must be manually indexed, or the objects within the images must be automatically recognized and appropriate keywords added to the index. Whether automatic or manual indexing is used, the descriptive words added by the indexer may or may not conform to the keywords employed by the user. One approach utilizes two different keyword indexing systems, conceptual keyword and scene description keyword, to retrieve images from a database. However, the developed techniques for indexing and retrieval are abundant, but there are no universally accepted techniques for feature extraction, indexing, and retrieval (Deb & Zhang, 2004).

Currently, new methods are emerging and being implemented for retrieving multimedia data objects (e.g., audio and video) based on their content. Research of content-based retrieval of multimedia information started in the early '80s with investigations into the retrieval of static images, which in turn was based on pattern recognition research of the '70s. Research into the retrieval of images from video has received more attention during the '90s. The approaches to aiding the retrieval of visual images from graphics or videos can be classified into the following categories:

- Keyword, in which the content of the images is described by an indexer using keywords or a textual abstract.
- Content-based image retrieval of features that can be automatically extracted from images. The features are selected according to certain criteria, such as color, texture, or shape; by allowing the user to sketch images and then retrieve similar images from the database; by discerning the motion path of an object; and by identifying one object from its position relative to another. There are a number of commercially available systems, such as IBM's QBIC and Virage products, which are based on the retrieval of images using their non-semantic features.
- Concept-based retrieval in which semantic interpretation of the objects is added, such as identifying an object as a named person, a type of vehicle, etc. There are a number of systems in this category

such as Infoscope and WebSEEK. WebSEEK is a semiautomatic system for retrieving, analyzing, categorizing, and indexing visual images from the World Wide Web.

Many algorithms have been focused on content-based image indexing, but researchers have been investigating an indexing system based on the spatiotemporal characteristics of objects that appear in multimedia applications. Spatial positions are represented by two-dimensional coordinates while temporal relationships are represented by a single dimension, time.

Other systems have been developed for automatically indexing video streams received in real time. The source of the video may be, for example, television broadcasts or security cameras. The incoming video data stream is passed to an event detection module that detects scene changes, significant audio changes, camera operations, and the motion of objects in the video. The event detection module uses this information to define the boundaries of what the authors refer to as events. The event boundaries are specific to the application; in broadcast television these may be scene changes or the appearance of captions, and in security surveillance video the beginning of a new event may be signified by a new object entering the scene. The first frame of an event is used as its key frame. The key frame, the event itself (comprising both video and audio), and its time index will be stored for a certain period of time. The key frames provide a visual index for the viewer. The view of the index displayed can be changed to show a more global view, with an index that includes every key frame. Content-based queries are often combined with text and keyword predicates to get powerful retrieval methods for image and multimedia databases.

Hypermedia video links is another system for content-based retrieval of information from a video database. Raw video data is stored in the video storage. Video clips can be extracted from the video object database using video object software. Developers of hypermedia applications can also use the video object manager to retrieve and play video data objects. The video data structure for the hypermedia multimedia authoring environment is quite straightforward. Using the video object manager, the developer of an application can extract clips manually from a video to create primitive video data objects. Attributes, such as display size and frame rate, inherited from the video from which it is extracted can be associated with the clip or altered by the developer. Complex data objects can be created using combinations of primitive and existing complex video data objects. The clips within a complex video data object are arranged in the sequence in which they will be played back. The next section introduces known multi-



media management products, examples of applications using these products, and significant standards.

## KNOWN MULTIMEDIA MANAGEMENT PRODUCTS

Corporations such as IBM, Virage, Silicon Graphics, ARDA, Oracle, Autonomy, and ALPHATECH are major providers of multimedia database management, communication, and content management software for corporations, media and entertainment companies, universities, and government agencies worldwide.

The IBM's CueVideo project provides technologies to automatically summarize and index videos and to make them much easier to browse. One approach is using audio stream for search and using video stream for quick browsing in a complementary manner to provide the desired video search functionality (Brown et al., 2001). CueVideo is an ongoing research project to address the challenges of large video databases.

One example of a mobile multimedia application is MobiDENK (Krosche, Baldzer, & Boll, 2004) for monument conservation. The Hermitage museum's Web site uses an IBM product, QBIC (Query By Image Content), for searching archives of world-famous art. QBIC is a system developed by IBM to explore various content-based retrieval methods. Queries in QBIC can be based on color and texture patterns, user-drawn sketches, example images, camera and object motion, or other graphical information, such as the color balance; e.g., retrieve all images with 30% blue and 10% yellow colored pixels. The QBIC database can handle both still images and video clips. In common with other systems, videos are divided into shots, using techniques similar to those described above. One frame from each clip is extracted or generated as a representative frame for the clip. In a motion sequence there may not be a single frame that is representative of the entire shot. In this case, QBIC constructs an image of the complete background from the sequence of frames and on to this superimpose images of the foreground objects.

QBIC's capabilities are available in DB2 Image Extenders, which are components of IBM's scalable, multimedia, Web-enabled DB2 Universal Database. IBM developed different multimedia retrieval applications for the DB2 Universal Database as follows:

- DB2 Audio Extender enables audio retrieval.
- DB2 Image Extender enables image retrieval. Visual features such as color and texture patterns are used for search criteria. For example, a photographic database may be queried for thumbnail

images of all pictures stored in GIF format, and the name of each picture's photographer can be listed. Then, Image Extender invokes a browser.

- DB2 Video Extender enables video retrieval. Video and traditional business data may be included in a single query.

For example, Image Extender is used to store print ads, Audio Extender for broadcast ads, and Text Extender for ad scripts. The user can retrieve all the objects in a single query and then preview video ads as video storyboards using the Video Extender.

Hillsborough County uses Virage's products to create a central digital archive of government proceedings, which automates a formerly manual workflow and provides real-time access to content. Virage products include capabilities as follows:

- IDOL provides the infrastructure for capturing, feeding, and delivering rich media through the enterprise.
- VS Archive is a software solution used by enterprises to store, categorize, manage, retrieve, and distribute video, audio, and other rich media content. It includes new features such as advanced conceptual retrieval and automated hypertext linking to enterprise information found in more than 300 different data types.
- VS News Monitoring is a real-time monitoring and content management solution used by enterprises and government agencies to automatically track content for time-sensitive, strategically significant events. New features within VS News Monitoring include real-time data access and advanced concept-based retrieval. Concept-based retrieval is especially important for monitoring and searching news feeds in foreign languages, because users may work in second languages or rely on transcriptions that may contain misspelled words. With IDOL, VS News Monitoring not only proactively alerts users to broadcast news but also delivers related internal content as well as information from Web sites.
- VS Webcasting is an enterprise software solution that simplifies and streamlines the entire workflow for producing live Webcast events and on-demand or rebroadcast presentations for large audiences.

Silicon Graphics and Virage combine products to create breakthrough media management systems. For example, Virage Media Management System and the Silicon Graphics StudioCentral asset management system enable companies to automatically and intelli-



gently catalog large libraries of videotape and multimedia content into a compact, online database. The combined products provide users with a complete media management system to find and manage their media through a simple Web browser. The next section discusses main standards for multimedia applications.

### MAJOR STANDARDS

Many aspects of the multimedia information life cycle are affected by regulatory compliance (Golshani, 2004), and requirements for latest object database management development standards are proposed (Cardenas, Pon, Panayiotis, & Hsiao, 2003). MPEG-7 and MPEG-21 standards have been essential for the development of multimedia applications. The MPEG-7 standard, formally named Multimedia Content Description Interface, provides a rich set of standardized tools to describe multimedia content. Both human users and automatic systems that process audiovisual information are within the scope of MPEG-7. MPEG-7 offers a comprehensive set of audiovisual description tools (the metadata elements and their structure and relationships, which are defined by the standard in the form of descriptors and description schemes) to create descriptions (i.e., a set of instantiated description schemes and their corresponding descriptors at the user's will), which form the basis for applications enabling efficient access (search, filtering, and browsing) to multimedia content (Chang, Kikora, & Puri, 2001).

MPEG-21 has established a work plan for future standardization. Nine parts of standardization within the multimedia framework have already started.

### FUTURE TRENDS

Content-based retrieval is a very active area of research despite the advances that have been made. Recently, Virage announced its participation in the Video Analysis and Content Extraction (VACE) media communication and content program sponsored by ARDA. The research project will be focused on the development of video content extraction technology. The initial focus of the VACE program is to develop automatic detection and recognition technologies from various video-related sources, including indoor and outdoor activities, unmanned air vehicles, and television news broadcasts. Over time, VACE technologies aim to provide significant improvement in indexing and retrieval, understanding, image processing, data mining, filtering and selection, and storage and forwarding mechanisms. In addition,

research is occurring on new compression techniques, such as wavelet, vector, and fractal methods, to ensure multimedia delivery with high quality.

In the future it should be possible to automatically recognize and identify objects that appear in still images and video. To achieve this, it will almost certainly require significant developments in the application of artificial intelligence techniques to multimedia database systems. These developments will lead to automatic recognition and indexing of video footage and still images and result in the development of a wide range of applications relying on the content-based retrieval of multimedia data objects.

### CONCLUSION

Multimedia is becoming present everywhere: at home, business, school, hospital, road, etc. Digital media has been acknowledged as a standard data type, allowing for increased personal communications, business-to-employee, business-to-business, and business-to-consumer applications. These applications require complex and large multimedia databases, causing changes in computing and solution architectures to be maintained by organizations. In addition, technologies pertaining to communication, coding, compression, content distribution, storage, mobile computing, media servers, cryptology and watermarking, and digital media management will grow in order to address solutions for multimedia services. While creating digital media is not expensive, it is generally expensive to manage and distribute it. Browsing large multimedia databases can become complex and will demand faster and more efficient algorithms for indexing and retrieval of structures. In addition, multimedia distribution in a mobile computing environment will continue to be the center of research for next-generation multimedia systems.

### REFERENCES

- Bourbakis, N. (2004). Digital multimedia on demand. *IEEE Multimedia*, 11(2), 14-15.
- Brown, E. W., Srinivan, S., Coolen, A., Ponceleon, D., Cooper, J. W., & Amir, A. (2001). Towards speech as knowledge resource. *IBM Systems Journal*, 40(4), 985-1001.
- Cardenas, A. F., Pon, R. K., Panayiotis, A. M., & Hsiao, J.-T. (2003). Image stack stream viewing and access. *Journal of Visual Languages & Computing*, 14(5), 421-441.

Chang, S.-F., Sikora, T., & Puri, A. (2001). Overview of the MPEG-7 standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6), 688-695.

Deb, S., & Zhang, Y. (2004). An overview of content-based image retrieval techniques. *18<sup>th</sup> International Conference on Advanced Information Networking and Applications (AINA'04)* (Vol. 1, pp. 59-64).

Dunckley, L. (2003). *Multimedia databases: An object-relational approach*. London: Addison-Wesley.

Golshani, F. (2004). Multimedia information lifecycle management. *IEEE Multimedia*, 11(2), 1.

Krosche, J., Baldzer, J., & Boll, S. (2004). MobiDENK—Mobile multimedia in monument conservation. *IEEE Multimedia*, 11(2), 72-77.

Shih, T., & Wang, P. P. (2004). *Intelligent virtual world: Technologies and applications in distributed virtual environment*. Hackensack, NJ: World Scientific.

Shu, W., & Yu, M.-Y. (2004). Resource requirements of closed-loop video delivery. *IEEE Multimedia*, 11(2), 24-37.

Thuraisingham, B. (2001). *Managing and mining multimedia databases*. Boca Raton, FL: CRC Press.

Tsai, C.-F., McGarry, K., & Tait, J. (2003). Using neuro-fuzzy techniques based on a two-stage mapping model for concept-based image database indexing. *IEEE Fifth International Symposium on Multimedia Software Engineering (ISMSE'03)*, 1, 6-12.

Almaden (n.d.). Retrieved July 22, 2004, from <http://www.almaden.ibm.com/projects/cuevideo/shtml>

IBM (n.d.). Retrieved July 25, 2004, from <http://www.qbic.almaden.ibm.com>

Infoscope (n.d.). Retrieved July 25, 2004, from <http://www.infoscope.com>

NASA Earth Observatory, <http://earthobservatory.nasa.gov:8000/Laboratory/index.html>, accessed July 23, 2004.

Oracle, <http://www.oracle.com>, accessed July 20, 2004.

TeraText, <http://www.teratext.com.au>, accessed July 23, 2004.

Virage, <http://www.virage.com>, accessed July 26, 2004.

WebSeek, <http://www.ee.columbia.edu/~sfchang/research>, accessed July 22, 2004.

## KEY TERMS

**Content-Based Retrieval:** Method for automatic multimedia content features extraction.

**Feature:** An attribute derived from transforming the original multimedia object by using an analysis algorithm; a feature is represented by a set of numbers (also called feature vector).

**Feature Extraction:** Use of one or more transformations of the input features to produce more useful features.

**Feature Selection:** Process of identifying the most effective subset of the original features.

**Indexing:** Mechanism for sorting the multimedia data according to the features of interest to users to speed up retrieval of objects.

**Metadata:** Information about multimedia data objects, applications, processing, and delivery requirements.

**Multimedia Database:** A repository of different data objects such as text, graphical images, video clips, and audio.

# Multiparticipant Decision Making and Balanced Scorecard Collaborative

**Daniel Xodo**

*Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina*

## INTRODUCTION

The building of *decision support system* (DSS) for integrated workgroups sets out problems related to the nature of the decision process as well as its operative implementation.

The multiparticipative decision systems (MDM) abound in the management of companies and public/private institutions. In these companies, the partial decisions of a group of people in their responsibility field reciprocally influence and determine the operation of the system. The achievement of the strategic goals requires unit of vision and interpretation of those objectives.

The *balanced scorecard* (BSC) is a tool for unifying the vision and allows a common element to interpret reality, and the environment changes for a heterogeneous group that has to decide among the multiple possible alternatives of a strategic plan.

It is necessary to have at one's disposal the knowledge and the integration of information and decisions through models of collaboration and integration.

This research work analyzes the usage of *BSC* as a link element between many decision makers, having different responsibilities and perspectives, who share strategic goals.

This usage requires becoming efficient in an accurate integration of:

- a. Collaborative-task environment
- b. Shared access to databases
- c. Techniques to get new information (relationships, indicators, ratios) such as *OLAP*, *datawarehouse*, and *data mining*
- d. Decision makers constant learning (from the obtained results)

In these models, it is possible to play down the interpretative divergence of the personal differences of the decision makers.

The analysis of the applications, decision models, integration and use of these systems *computer-supported cooperative work* (CSCW) constitutes a prosperous field to develop management computerized structures.

To integrate the functions of a company into a unified strategy, work models facilitate the spread of knowledge. BSC is much more than an information system; it is a knowledge management system which allows adapting, in a coordinate way, those decisions leading to the achievement of strategic goals. Thus, integrated techniques, tools, and working methodologies will also make possible:

- a. A rise in shared models
- b. A rise in learning speed
- c. Improvement in the quality of decisions
- d. Major action coordination

In these models, it is possible to play down the interpretative divergence of the personal differences of the decision makers.

## BACKGROUND

### Multiparticipant Decision Making

We can define it as "an activity conducted by a collective entity composed of two or more individuals and characterized in terms of both the properties of the collective entity and of its individual members". Each of the MDM types has a specific structure of interaction among the decision makers and the participants (Marakas, 1997)

The MDM is a common and growing reality. Diverse modern business structures such as virtual companies, the strategic alliances, the "cooperation-competence" relations, and the new philosophies and management modes require forms, methodologies, and techniques applied to the decisions that surpass space and time limitations.

This growing development of management concepts significantly collaborates informatics. The aforementioned limitations imply others which are the ones that harm the quality, swiftness, and comprehension of decisions such as:

- a. Shared mission
- b. Unified vision
- c. Availability of information

- d. Criteria unification
- e. Acceptation of particular aims
- f. Coordination
- g. Feedback (awareness of the effects and the decisions)

The different types and classes of support technologies show the different alternatives and possibilities of improving quantitative tools and informatics. Those decisions should be shared by many people.

BSC offers the possibility of joining them according to their application opportunities without detriment to the utilities of those techniques and tools, in a cause-effect analysis relative to the tangible and intangible components of the strategic decisions.

### Scheme of Decision Processes

Simon (1960) elaborated a model about the problem-solving characteristics, which represents a permanent scheme of the decision-maker's activity (Figure 1).

### Decisions Based on Strategy

These decisions are of particular interest in our case since we can consider the strategy as Byrnes and Chesterton (1978) and Mintzberg (1993) did:

- a. Plan
- b. Action rules
- c. Behavior pattern
- d. Position
- e. Perspective

Since these are strategies useful to make decisions on tactic plans and to establish actions consistent with those ones, it is of particular importance that their comprehension, alignment, and participation of those who are responsible for carrying it out (Davenport & Prusak, 1997; Steiner, 1997).

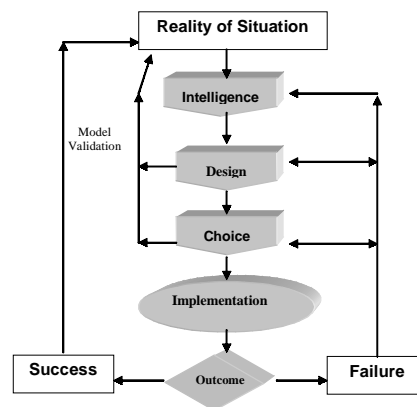
*Comprehension and involvement need an "integrated knowledge environment" to elaborate and apply the chosen strategies.* (Tissen, Andriessen & Lakanne, 2000)

### Factors, Conditions, and Perspectives Affecting Decisions in Organizations

#### Factors

Decisions in an organization are affected by factors characteristic of the relationships established within the organization and with its environment, such as in Marakas (1999) and Holsapple and Joshi (2001):

Figure 1. Scheme of decision processes



- a. Group structure
- b. Roles
- c. Processes
- d. Decision styles
- e. Norm and decision rules

These characteristics have non-quantitative main components; then it is necessary to get a hold of the right conceptions for their coordination and treatment, orientated to optimize team work decisions and to improve those relationships which help to encourage group synergy (Fleischer & Mahaffy, 1997).

### Group Decision-Making Conditions

Group decisions are taken under conditions of limited knowledge, uncertainty, poor communication, option unawareness, and diverse goals (and sometimes contradictory) of the decision makers.

The difficulties which generate the conditions previously mentioned tend to increase considering the different levels (operative, tactic, and strategic) in which they have to be implemented (DeSanctis & Gallupe, 1987).

Modern *knowledge management systems* are supposed to meet different decision makers abilities as they face common problems and shared decisions. BSC comes out as a model gathering different techniques, perceptions, and approaches through perspectives shared by all the group members in strategic planning adaptable to the changing conditions of the environment and the organization itself (Pigott, 2000). These models represent a sort of "virtual community", qualified by knowledge of the reality towards a specific strategic goal (Tissen et al., 2000).

## Decision Perspectives

Keen and Morton (1978) have stated a classification of the perspectives of a decision-making process which shows the extent and the transcendence of its improvement:

- a. Administrative rationale perspective
- b. Processes orientation perspective
- c. Organizational procedures perspective
- d. Political perspective
- e. Individual difference perspectives

All of the decision perspectives have the characteristic of *information* and *knowledge*.

As *information* because they are a series of formal rules such as descriptions of processing, probabilities of occurrence, and representations of meaning by means of symbols or mathematic operations.

As *knowledge* because they are related with description, analysis, comprehension of the environment, and any probable change necessary for organization strategies.

Knowledge in organization bears the following characteristics (Barnes, 2002):

- a. Comprehension and reflection
- b. Communication and broadcasting
- c. Knowledge aspects and results

## CRITICAL ISSUES AND PROBLEMS

### Decisions and Knowledge

As knowledge has its source in learning, it broadens considering three main components:

- a. Cognitive learning
- b. Social learning
- c. Feedback learning

As an exponential factor of decisions intelligibility ( $I^3$ ) (Tissen et al., 2000).

Each learning component has an important part of shared knowledge, the three of them a consequence of interaction and mutual reinforcement in understanding reality and its problems.

Basing in these concepts to generate (make) shared decisions means a change in the culture and attitudes which require self/own elements to materialize in conducts and results (Neidorf, 2002).

## Knowledge Management Systems

*A knowledge management system (KM) should be useful to whoever is implicated in the comprehension processes, evaluation and reorganization of the company.* (Frank, 2002)

This means offering relevant knowledge to groups conformed by executives, analysts, technicians and assessors, employees, customers, and suppliers who take part in the processes.

Knowledge management systems must fulfill the following requisites (Frank, 2002):

- a. Conceptual level emphasis
- b. Existing knowledge reuse
- c. Individual needs adaptation
- d. Intuitive comprehension
- e. Different perspectives
- f. Perspectives integration

One of the main challenges of the knowledge management systems is the transformation of implicit knowledge (subjective, hard to recognize and transmit) in explicit (one/knowledge) and its transference and integration (Nonaka & Takeuchi, 1995).

Beyond conceptual requirements, KMs must have technical features which make them useful to decision groups, such as in Alavi and Leiner (1999), KPMG (1998), and Easterby-Smith (1997):

- a. Information capable of turning into actions
- b. Categorized data for usage
- c. Information filters
- d. Information accessibility
- e. Data storage systems (*data warehouse*)
- f. Integrated databases
- g. Extraction information techniques (*data mining*)
- h. Analytical decision techniques
- i. Facilitate constant learning
- j. Facilitate group and organizational learning
- k. Advance information and communication technologies

### BSC as Knowledge Management System

The possibility of getting the previously mentioned learning modes greatly depends on the utilization techniques and tools which enable getting hold of data-generated knowledge and make use of it from an interpretative level or state of reality.



Balanced Scorecard Collaborative (BSC) is basically a strategic management system which allows by means of cause-effect relationships and the indicators which represent them (Kaplan & Norton, 1997):

- a. Clarified and translate or transform vision and strategy
- b. Communicate and link strategic objectives and indicators
- c. Plan, establish objectives, and align strategic initiatives
- d. Increase feedback and strategic formation

For each perspective, the executors have quantification elements of the relevant variables and the possibility of deepening their analysis by means specific applications, having at the same time a global view of the strategy and the degree of general realization.

Some important aspects in BSC as knowledge management systems are those of principal utility in the elaboration of group decisions.

The data and information expressed may be cleverly interpreted and used for the decision by different people from different points of view:

- a. Its offers unified vision
- b. It facilitates knowledge interchanging
- c. Promotes learning in different levels and ways

### Informatic Requirements in Applying BSC in MDM

#### Virtual Collaborative Environment

The overcoming of these limitations is especially favored by the implementation of *Virtual Collaborative Environments* (EVC) where the application of management technologies plays a vital role.

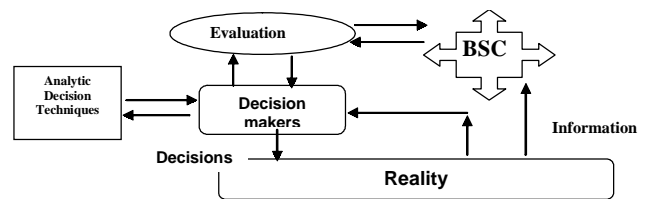
*Groupware* and *workflow* techniques belong to that framework.

The combination of these technologies with strategic management tools, such as BSC, can provide the shared knowledge conditions, feedback, and adapted to the approaches heterogeneity, capacities, and interests of the integrants of the decision-making group.

Network development (*Internet, Intranet, Extranet*) opened new possibilities to organization management and informatics usage. The tools previously mentioned, fundamental in collaborative task environment, are included in this category.

*Business Process Component Model* (BPCM) shows the link reality-information-evaluation-decisions, incorporating a BSC for the analysis (Figure 2).

Figure 2. BSC in BPCM



The model contains

1. Communication
2. Collaboration
3. Coordination

which are requirements of a system of decision generating in the business processes. The model will require a group of interfaces and formats to define its operation from a chosen strategy.

### Shared Strategy Control Model

Different BSCs can be applied to the integrant parts of a general strategy which joins and controls them in a shared BSC (Figure 3) (Kaplan & Norton, 2001).

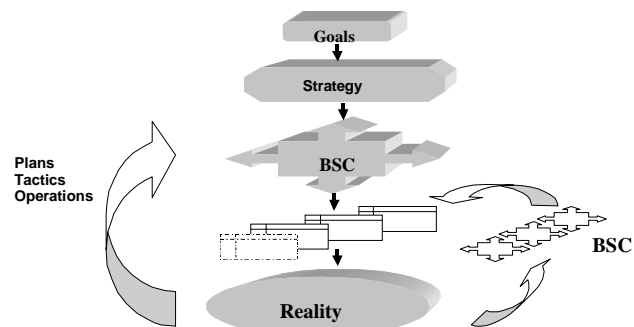
### Group Decision Support Systems (GDSS)

They are interactive systems based on computers to sort out non-structured problems by a group of decision makers.

A GDSS provides substantial improvements in the operation of decision groups such as in Landon and Landon (1996):

- a. Increase the precision, knowledge, and effectiveness of the planning

Figure 3. Strategy and BSC



## Multiparticipant Decision Making and Balanced Scorecard Collaborative

- b. Increase the participation and the integration
- c. It is a collaboration tool
- d. It is useful to generate ideas
- e. Increase the objectivity of the evaluation
- f. Establish priorities
- g. Improve the organization
- h. Generate and increase the availability of necessary documentation
- i. Facilitate access to external information
- j. Promote institutional development
- k. Reconcile the concepts and information used by the integrants

Apart from the great elaboration and complexity models, among the most used in GDSS are:

- a. Electronic questionnaires
- b. Benchmarking tools
- c. Organization of ideas
- d. Vote techniques or priorities
- e. Leader identification and analysis tools
- f. Policy formulation tools
- g. Group dictionaries
- h. Electronic meeting systems

As it can be appreciated, the possibilities of achieving an effective system of interaction leading to the best decisions which are wide and varied.

### GDSS in Network Structures

Participative models using BSC can also be used in business networks. A clear example is the linked tourist services (transportation, lodging, shows, excursions, etc.).

These networks need appropriate information processing systems for fluidity, flexibility, and originality of the decisions (Serra & Kastika, 1994). The dynamics of information, the cooperation for achieving results, and the permanent adaptability to the market requirements need tight links of knowledge and management.

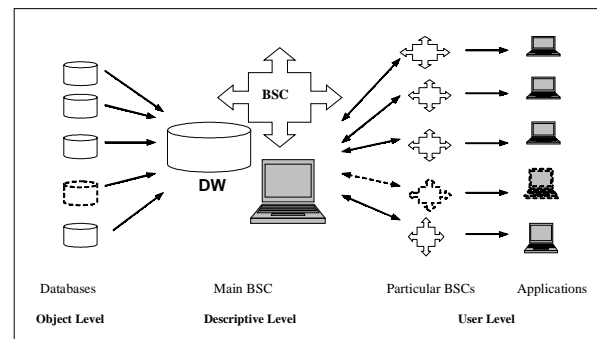
A model with an adaptable structure to the requirements and the development of competence cultural guidelines which allows to establish global shared strategies can be set up over these networks.

In these common strategies, the specific ones which belong to each participant in the network will be inserted.

### Binding Scheme of the Group Decision Support System (GDSS)

A possible architecture for multiparticipative decisions through knowledge management (Figure 4) shows the

Figure 4. Architecture for multi-participative decisions



characteristics of the system which has the following properties (Abecker et al., 2002):

- a. Active help in the search and detection of information
- b. Integration functions (data, formal knowledge, representation, and knowledge materialized in products and processes)
- c. Self-adaptation and self-organization

### BSC Advantages and MDM Use

To the already mentioned characteristics of BSC and GDSS as specific of the applications, the advantages originating in the functional usefulness of their combination should be added.

In a non-exhaustive enumeration, the following points could be mentioned:

1. Regular monitoring of the results of the strategy expressed on the BSC.
2. Immediate knowledge of the consequences and individual-group actions (Malina & Selto, 2001).
3. Detection through the established indicators of the difficulties inconsistencies, and diversions, and other flaws of the chosen strategy (Kaplan & Norton, 1997).
4. Possibility of detecting, through the application of Data mining and data warehouse strategies, new indicators that evidence unconsidered relations the moment of setting out a strategy. The development analysis and individual applications through a shared database in which each participant can evaluate decision field and integration in the context of development of the strategy.
5. It allows evaluation of intangibles with conceptually homogeneous criteria, incorporating them as a principal factor in strategic management. This idea

is particularly important when the group includes members with different capacities, interests, or activities (Miyake, 2002).

6. It determines a collaborative learning system (Keen, 1991).
7. Group and individual learning are increased by the multiplicity of perspectives and approaches and their integration in the information treatment (Shih-Jen & McKay, 2002).
8. Speeds up the feedback for the organization and its members (Olve, Roy & Wetter, 1999).
9. Stimulates the “network” effect that benefits the “open systems” (Thierauf, 1999).
10. Integrates the decision makers in a unified process with others that without being part of the organization or company have a bearing in the results of the strategy. Hence, the external influences are systematically considered and not only when its accidental bearing modifies the strategic results (Malina & Selto, 2001).
11. Defines a systemic model of strategic control (Ballvé, 2000).
12. Facilitates the disposition to share risks and rewards (Hamel, 2000).
13. Clarifies the purposes of the participants since it expresses them in the reference indicators.
14. Improves the strategies scope, broadcasting its knowledge and understanding among the members of the organization (Miyake, 2002).
15. Increases the specialization of workers, businesspeople, analysts, and executives (Moore, Row & Widener, 2001).
16. Generates great knowledge and domain of strengths, opportunities, weaknesses, and threats.
17. Generates a style of articulation in groupwork, exceeding organizational structures.
18. Establishes standards of qualities benefiting the application of techniques such as benchmarking.
19. Admits the possibility of applying simulation models to generate and analyze alternatives.
20. Defines and holds ethical conceptions in management organization (Fainstein, 1997).
21. Benefits the integration of the established goals.
22. Allows rational integration of personal goals to the general goal.
23. Reduces internal contradictions in organizations (HR Focus, 2002).
24. Enlarges motivation through learning and knowledge.
25. Increases the morality of work and cooperation, strengthening bonds and the shared satisfaction of something done well.

## **FUTURE TRENDS**

The development and application of BSC in decision-making groups foresees and opens interesting fields to study its structures and functioning models. Some relevant topics arise (Barnes, 2002; Bohn, 1994; Frank, 2002; Kaplan & Norton, 2001; Neidorf, 2002; Probst, Raub & Romhardt, 2000):

- a. Develop system architectures for MDM decisions
- b. Analyze information structures as regards comprehension levels, analysis and interpretation levels, and decision levels
- c. Represent information in multi-layer systems
- d. Develop dynamic indicators
- e. Develop scalability of indicators
- f. Integrate organizational memory of the company

## **CONCLUSION**

In this way, the modes of participation in the decisions are enriched with tools that permit reaching agreement and making negotiations possible in relation to the shared knowledge from diverse perspectives.

These kinds of work will create new dynamics and concepts to develop decision models.

Together with the relevance that new computing technologies have, there are aspects characteristic of the applications of the methodology and the relations which are developed as a consequence of its usage.

Available informatic technologies power management models and open possibilities to new developments in administrative sciences.

In the new management models, substantial improvements related to technical and ethical aspects are evidenced.

Technically, the high increase of the available information, selection, storing, formal elaboration (quantitative and qualitative), and generated knowledge application possibilities.

Ethically, modifications to behaviors that, as a consequence of general knowledge, will be reflected in new consensus and responsibility mechanisms, derived from the increase in participation in organizational decisions.

## **REFERENCES**

Abecker, A., Bernardi, A., Hinkelmann, K., & Sintek, M. (2002). Infraestructuras de información empresarial para la entrega activa de conocimiento sensible al contexto.

## Multiparticipant Decision Making and Balanced Scorecard Collaborative

- In S. Barnes (Ed.), *Sistemas de gestión de conocimiento: Teoría y práctica* (pp. 175-219). Madrid, Spain: Thomson.
- Alavi, M., & Leiner, D. (1999). Knowledge management systems: Emerging views and practices from the field. *Proceedings of the Hawaii International Conference on Information Systems*, Maui, Hawaii.
- Andreu, R., Ricart, J., & Valor, J. (1996). *Estrategia y sistemas de información*. Madrid: McGraw-Interamericana de España.
- Ballvé, A. (2000). *Tablero de control*. Buenos Aires: Ed Macchi.
- Barnes, S. (2002). *Knowledge management systems: Theory and practice*. Thompson Learning.
- Bohn, R.E. (1994). Measuring and managing technological knowledge. *Sloan Management Review*, 36(1), 61-73.
- Byrnes, W., & Chesterton, B. (1978). *Decisiones y estrategia*. Buenos Aires: El Ateneo.
- Chaffey, D. (1998). Groupware, workflow and intranets. *Reengineering the enterprise with collaborative software*. Digital Press.
- Davenport, T., & Prusak, L. (1998). *Working knowledge*. Boston: Harvard Business School Press.
- DeSanctis, G., & Gallupe, R. (1987). A foundation for the study of group decision support systems. *Management Science*, 33(5), 589-609.
- Easterby-Smith, M. (1997). Disciplines of organizational learning: Contributions and critiques. *Human Relations*, 50(9), 1085.
- Fainstein, H. (2000). *La gestión de equipos eficaces*. Buenos Aires: Ed Macchi.
- Fleischer, C., & Mahaffy, D. (1997). A balanced scorecard approach to public relations management assessment. *Public Relations Review*, 23(2), 117-143.
- Frank, U. (2002). Knowledge management systems: Theory and practice (pp. 115-131). Thomson Learning.
- Hamel, G. (2000). *Liderando la revolución*. Barcelona: Harvard Business School-Gestión.
- Holsapple, C., & Joshi, K. (2001). An investigation of factors that influence the management of knowledge in organizations. *Journal of Strategic Information Systems*.
- HR Focus. (2002). How to blend learning and knowledge management. *Institute of Management & Administration*, 79(7), 5.
- Kaplan, R., & Norton, D. (1997). *The balanced scorecard: Translating strategy into action*. Boston: Harvard Business School Press.
- Kaplan, R., & Norton, P. (2001). Transforming the balanced scorecard from performance measurements to strategy management. *Accounting Horizons*, 15(1), 87.
- Keen, P. (1991). *Shaping the future: Business design through information technology*. Boston: Harvard Business School Press.
- Keen, P., & Morton, S. (1978). *Decision support systems: An organizational perspective*. New York: Addison-Wesley.
- KPMG. (1998). Knowledge management. *KPMG Research Report*.
- Landon, K. & Landon, J. (1996). *Administración de los sistemas de información*. Mexico: Prentice Hall-Hispanoamérica.
- Malina, M., & Selto, F.H. (2001). Communicating and controlling strategy: An empirical study of the effectiveness of the balanced scorecard. *Journal of Management Accounting Research*, 47-90.
- Marakas, G. (1999). *Decision support systems*. NJ: Prentice Hall.
- Mintzberg, H., and Quinn, J.B. (1993). *El proceso estratégico. Conceptos, contextos y casos* (2nd ed.). Mexico: Prentice Hall.
- Miyake, D. (2002). Beyond the numbers: After years of evolution, balanced scorecard applications now integrate strategy and management for competitive advantage. *Intelligent Enterprise*, 5(12), 24-30.
- Moore, C., Rowe, B.J., & Widener, S. (2001, November). HCS: Designing a balanced scorecard in a knowledge-based firm. *Issues in Accounting Education*, 16(4), 569-601.
- Neidorf, R. (2002, September-October). Knowledge management: Changing cultures changing attitudes. *Online*, 26(5), 60-62.
- Nonaka, I., & Takeuchi, H. (1995). *The knowledge-creating company*. Oxford: Oxford University Press.
- Olve, N.G., Roy, J. & Wetter, M. (1999). *Performance drivers: A practical guide to using the balanced scorecard*. New York: John Wiley & Sons.

Pigott, S. (2000, May). Knowledge management systems for business. *Business Information Alert*, 12(5), 11.

Probst, G., Raub, S., & Romhardt, K. (2000). *Managing knowledge*. Chichester, UK: John Wiley & Sons.

Serra, R., & Kastika, E. (1994). *Re-estructurando empresas*. Buenos Aires: Ed Macchi.

Shih-Jen, K.H. & McKay, R.B. (2002, March). Balanced scorecard: Two perspectives. *The CPA Journal*, 72(3), 21-25.

Simon, H.A. (1960). *The new science of management decisión*. New York: Harper and Row.

Steiner, G. (1997). *Strategic planning*. Mexico: CECOSA.

Thierauf, R.J. (1999). *Knowledge management systems for business*. Quorum Books.

Tissen, R., Andriessen, D., & Lakanne, F. (2000). *The knowledge dividend*. New York: Prentice Hall.

## KEY TERMS

**Balanced Scorecard Collaborative:** It is a strategic management system that measures, by means of quantitative relations of different selected variables, the behavior of the organization taking into account the settled aims which are established in different perspectives (Increase, Internal Processes, Customers, Finances). The analysis is based on the cause-effect relations between the variables and ratios that represent them.

**Benchmarking:** Procedure to compare and improve the manufacturing quality and services based on the comparison of operations, methods, procedures, and processes inside and outside the organization.

**Cognitive Learning:** It is a consequence of the vision of the situation in the light of a new aspect that enables the comprehension of logic relations or the perception of relations between means and aims.

**Feedback Learning:** It deals with learning based on the input-process-output-feedback process in which three

laws shape the learning process: exercise law: reiteration strengthens the connection between response and stimulus; effect law: the succession of stimulus-response is not enough for learning; reinforcement is needed; and disposition law: achieving goals is a reinforcement particular to every action that has a clear aim.

**GDSS:** A collective of computer-based technologies that are specifically designed to support the activities and processes related to multiparticipant decision making.

**Group Decisions:** Decisions adopted by a group of people with complete interaction under majority or consensus conditions.

**Groupware:** Informatic technologies which allow a group of people to work on a common task by giving them a shared-environment interface (Chaffey, 1998). Its main features are interaction among users (video, text, and sound); centralized and shared information; and groupwork conscience.

**Knowledge:** It is the application of a combination of instincts, ideas, rules, procedures, and information to guide the actions and decisions of a problem solver within a particular context.

**Knowledge Management (KM):** Set of activities which deals with knowledge acquisition, selection, internalization, and usage.

**Social Learning:** Learning based on attention, perception, and memory capacities is significantly influenced by the socialization and education context and particularly by language so as to create human knowledge.

**Virtual Collaborative Environments:** Computing applications that include systems of groupware in order to assist work groups with a common goal, where participants work in their own computers but share data and information by means of a user's interface.

**Workflow:** Systems which are meant to automatize and control business processes. Among its functions are task assignment, alerts, common tasks cooperation, align resources with the strategy, automatization of business processes, and tracking and oversight.



# Natural Language Front-End for a Database

**Boris Galitsky**

*Birkbeck College University of London, UK*

## INTRODUCTION

Whatever knowledge a database contains, one of the essential questions in its design and usability is how its users will interact with it. If these users are human agents, the most ordinary way to query a database would be in the natural language (Gazdar, 1999; Popescu, Etzioni, & Kautz, 2003; Sabourin, 1994). Natural language question answering (NL Q/A), wherein questions are posed in a plain language, may be considered the most universal but not always the best (i.e., fastest) way to provide the information access to a database. One should be aware that approaches to data access, such as visualization, menus and multiple choice, FAQ lists, and so forth, have been successfully employed long before the NL Q/A systems came into play. In the following, I discuss situations in which a particular *information access* approach is optimal. The five basic means to access (i.e., search) the data with respect to the search methodology is highlighted:

1. Looking through the data itself
2. Consulting the explicit enumeration of choices (e.g., menus, lists, combo boxes)
3. Structured language-based database querying
4. Keyword search
5. NL Q/A (as a front-end to a database or to unstructured data)

For many data access problems, 1 and 2 compete with 3 through 5 in terms of efficiency. Frequently, explicit browsing of the data or using intermediate steps is sufficiently convenient; however, 1 is good for a limited amount of data, 2 requires data structuring and is not flexible, and 3 is used for *fully structured knowledge* with relational links. Note that database querying may involve the other (i.e., non NL) means: 1, 2, and 4 to build a query that runs against the data source. The most powerful approach to *data management* seems to be 3, wherein the retrieval is easily and naturally combined with the update. Also, it does not require data modification to provide the NL search, thereby transitioning to 5.

Nowadays, in a majority of applications, the number of queries that are run is quite limited and can be initiated via a form. However, this will not likely be the case in the future for semistructured knowledge representations. Next I enumerate the situation in which the traditional form-

based query specification is appropriate (i.e., rather than NL front-end):

1. A homogeneous domain includes the description of objects, identified by their names (e.g., cars, flowers, people).
2. A domain is completely unstructured; semantic links between its entities and objects are nonsystematic.
3. A domain is oriented to professional users and includes specific terminology.
4. Domain structure is very clear and a user is closely familiar with it.
5. The domain itself is almost unstructured, but the objects of search fall into clusters in accordance to their features. A Boolean combination of keywords is then well-suited for the search of objects in such a domain.

## BACKGROUND

An NL front-end for a database implements query translation from NL to SQL. The difficulty in this task is that, first, NL is ambiguous and, second, its understanding requires domain knowledge that is not represented in a relational database (i.e., meanings of involved objects as words). The NL processing components that are required to achieve an accuracy desired under database querying are as follows (Galitsky, 2003; Gayatri & Raman, 2001; Wallace, 1984):

1. *Morphological and syntactic* analyses that produce links between words (i.e. a parser). This kind of analysis uses linguistic but not domain knowledge data.
2. *Semantic analysis* that establishes a mapping between some words or multiwords and table names, column names, and record values as well as SQL operators. Semantic analysis is based on the formal treatment of *meanings* of involved entities (Allen, 1995).
3. *Pragmatics analysis*, which evaluates the consistency of the obtained query against the database and then filters the hypotheses of syntactic and pragmatic analyses in case an input query is ambiguous.

In a traditional architecture of database front-end, the components include an *index*, a *lexicon*, and a *parser* (Adam & Gangopadhyay, 1997). The index is used to uniquely identify each form in the system through a conceptual representation of its purpose. The *form fields* specify database or nondatabase fields whose values are either entered by the user (i.e., user defined) or are derived by the form (i.e., system defined) in response to user input. A set of *grammar rules* is associated with each form. The lexicon consists of all words recognized by the system, their grammatical categories, roots, their associations (if any) with database objects and forms. The *parser* scans a natural language query to identify a form in a bottom-up fashion. The information requested in the user query is determined in a top-down manner by parsing through the grammar rules associated with the identified form.

The enumeration of the features for a database NL front-end (About.com; Beck, Mobini, & Kadambari; Gayatri & Raman, 2001) follows:

- **Independent domain creation:** Given a database, a system may be capable of automatically adjusting its syntactic and semantic unit to it, processing the names of columns and tables. Indeed, automatic creation of entities which correspond to lexical units and relationships between them is quite unreliable for a natural language front-end to a database. This is in contrast to a natural language interface to a collection of documents (automatic annotation), which is quite reliable.
- **Follow-up questions:** For example, if one initially asks, *Which store in California sold the most coffee in 1997?* she can then ask a follow-up question, such as, *Which one sold the most coffee?* and the NL front-end system will understand this to mean *Which store in California and In 1997 sold the most coffee.*
- **Specifying additional phrasings:** This is necessary so that syntactically different but semantically similar questions are converted into the same SQL query.
- **Maximum complexity of queries:** These can be measured as a number of entities in a query, assuming that they are interconnected. If a question can be split into a conjunction or disjunction of simpler questions, a user is expected to do it. For a database front-end, a query complexity can be measured as a number of tables mentioned.
- **Automatic superposition of syntactic and semantic templates:** If a NL system knows two entities (or a respective pair of phrasings) it can combine it online to represent a query. This is a quite desirable feature to increase the complexity of queries and also to merge various databases each having its own front-end.

## TECHNIQUE OF SEMANTIC HEADERS

The technique of semantic headers (SH; Galitsky, 2003) is intended to be the means of conversion of a relational database of abstract textual documents into a form, appropriate to be associated to a question and to generate an advice. There are two opposite common approaches to this problem. The first one assumes that complete formal representation of any textual document is possible, and the second one assumes that the textual information is too tightly linked to NL, and it cannot be satisfactorily represented without it. The former approach relies on the match of formalized query with the full-knowledge representation for answers, and the latter is based on the syntactic match between the question and sentences from answers. An important role of machine-learning-based technique for question answering is worth mentioning as well (Ng, Lai Pheng Kwan, & Xia, 2001).

The technique intermediate in respect to the degree of knowledge formalization. Only the data, which can be explicitly mentioned in a potential query, occur in semantic headers. The rest of the information, which would be unlikely to occur in a question but can potentially form the relevant answer, does not have to be formalized.

SH technique is based on logical programming, taking advantage of its convenient handling of semantic rules on one hand, and explicit implementation of the domain commonsense reasoning on the other hand. The declarative nature of coding semantic rules, domain knowledge, and generalized potential queries introduces logical programming as a reasonable tool. At the same time, the machinery of text annotation by the set of keywords has been proven to leverage the machine learning technique. Instead of using the keywords as semantic means to represent the meaning of a short textual document (answer), we use the logical formula where the keywords serve as atoms. Therefore, SH technique is a way of merging potential results of statistical approach to Q/A with the logical programming way of matching the formal representation of a query with the formal representation of an answer (semantic header of this answer). In legal domains, when the semantic of conversational language can only be ambiguously mapped into the semantic of the legal language, using just the statistical annotation by keyword does not lead to satisfactory results.

Consider the Internet auction domain, which includes the description of bidding rules and various types of auctions.

- **Restricted-Access Auctions:** This separate category makes it easy for you to find or avoid adult-only merchandise. To view and bid on adult-only items, buyers need to have a credit card on file

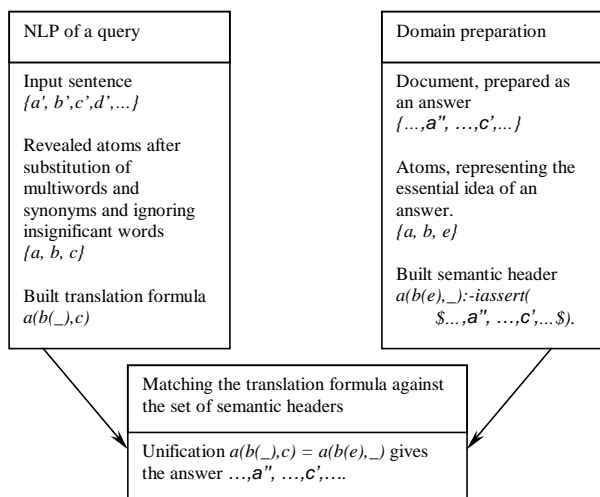
with eBay. Your card will not be charged. Sellers must also have credit card verification. Items listed in the Adult-Only category are not included in the New Items page or the Hot Items section, and currently, are not available by any title search.

What is this paragraph about? It introduces the “restricted-access” auction as a specific class of auctions, explains how to search for or avoid a selected category of products, presents the credit card rules, and describes the relations between this class of auctions and the highlighted sections of the Internet auction site. Rather than changing the paragraph to adjust it to the potential questions answered within it, consider all the possible questions this paragraph can serve as an answer to. Building the semantic headers of a textual document is based on the posing of a query understanding problem as the recognition of the best pattern (e.g. document, answer). For example, if there is a question such that the stated paragraph is a more appropriate answer than any other paragraph from the whole domain, then the stated paragraph should serve as the answer, or at least part of the answer.

Evidently, knowledge of the semantic model of the whole domain is required to build the set of semantic headers for a given paragraph. This paragraph serves to answer the following kinds of questions:

- What is the restricted-access auction?
- What kind of auctions sells adult-only items?

Figure 1. The information flow of the technique of semantic headers. The input query is subject to natural language processing (NLP, on the left). The answers are subject to the procedure of SH assignment, performed while preparing the Q/A domain (on the right). The essence of finding an answer is the unification of the translation formula with all the SHs of the domain.



- How do I avoid adult-rated products for my son?
- How does one sell adult items?
- When does a buyer need a credit card on file? Who needs to have a credit card on file? Why does a seller need credit card verification.

Below is the list of semantic headers for these answers.

```
auction(restricted_access, _):-restrictedAuction.
product(adult, _):-restrictedAuction.
seller(credit_card(verification, _), _):-restrictedAuction.
sell(credit_card(reject(\_, \_), \_), \_):-restrictedAuction.
seller(credit_card(\_, \_), \_):-restrictedAuction.
what_is(auction(restricted_access, \_), \_):-restrictedAuction.
```

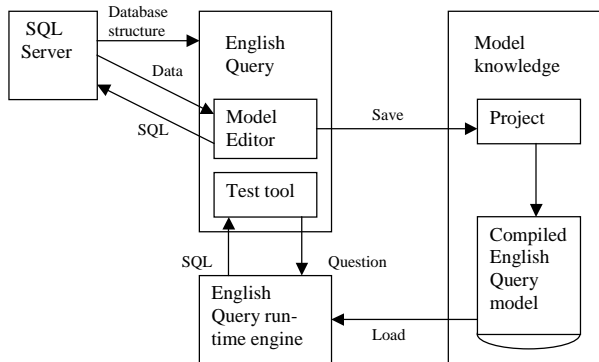
Then the call to *restrictedAuction* will add the stated paragraph to the current answer, which may consist of the multiple pre-prepared ones.

Matching the query translation with the totality of SHs is shown at Figure 1.

## FUTURE TRENDS

As to the particular implementation of a database NL front-end, we consider Microsoft English Query (About.com; Microsoft.), the integrated tool for building, adjusting, and deploying the SQL server database. Using English Query one can turn a relational database into an application, which provides end users with an option to pose questions in NL instead of forming a query with an SQL statement. The English Query project wizard allows users to automatically create an English Query *project* and *model* (see Figure 2). After the basic model is created, a developer can refine, test, and compile it into an English Query application and then deploy it (for example, to the Web). As to the database

Figure 2. The database structure (table names, field names, keys and joins) is incorporated into a project and a model



type of a Q/A domain, the knowledge and *semantic model* development tools are quite important.

Such a model contains all the information needed for an English Query application, including the database structure, or schema, of the underlying SQL database and the semantic objects (i.e., *entities* and *relationships*). A database front-end developer can also define properties for an application and add entries to the English Query dictionary as well as manually add and modify entities and relationships while testing questions and set other options to expand the model.

With the wizards, semantic objects are automatically created for the model. These include entities and relationships (with phrasings such as *customers buy products* or *Customer\_Names are the names of customers*). Entities are usually represented by tables and fields (see Figures 3 and 4).

An entity is a real-world object, referred to by a noun (i.e., person, place, thing, or idea), for example: *customers*, *cities*, *products*, *shipments* and so forth. In databases, entities are usually represented by tables and fields. Relationships describe what the entities have to do with

one another, for example, *customers purchase products*. Command relationships are not represented in the database but refer to actions to be executed. For example, a command to a compact disc player can allow requests such as “Play the album with song X on it.”

## CONCLUSION

The idea of natural language access to databases is neither a new idea nor a fundamentally different approach to information retrieval. NL front-end has been recently attempted for a video database (Gayatri & Raman, 2001). With over 3 decades of research and development into natural language processing (American Association for Artificial Intelligence, 1999; Pasca, 2003), it is still a hard task to provide information access through simple dialogues. For example, it would not seem too advanced for a user to ask simple questions such as Who was the founder of Oracle? or What is the distance to the moon? and get a simple, direct answer. These questions, though not very complex in structure nor ambiguous in phrasing,

Figure 3. Semantic objects of English Query. OLAP, online analytical processing, is accessible by natural language question answering as well.

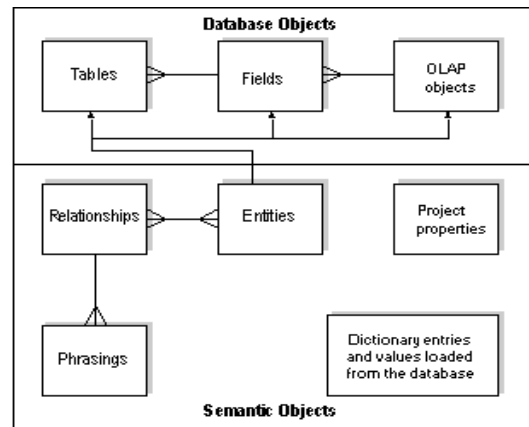
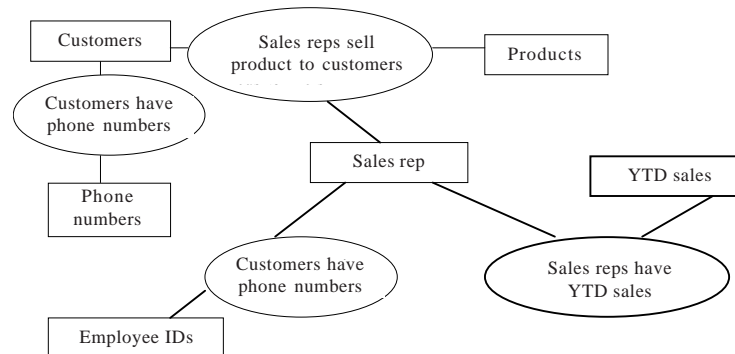


Figure 4. Relationships between entities





cannot be handled effectively by the current search techniques of the World Wide Web (Maybury, 2000, 2004). More intelligent search engines will break these queries into Boolean keyword searches to give back *founder* and *Oracle* or *distance* and *moon*, but others would require the user to formulate the queries into a formal language. Even so, keyword searches are known to be inaccurate and require the users to look through many answers, which point to articles that users must read rather than directly answer their questions.

## REFERENCES

- About.com. Microsoft English query. Retrieved June 15, 2004, from <http://databases.about.com/library/weekly/aa032402a.htm>
- Adam, N. R., & Gangopadhyay, A. (1997). A form-based natural language front-end to a CIM Database. *IEEE Transactions on Knowledge and Data Engineering*, 9(2), 238-250.
- Allen, J. F. (1995). *Natural language understanding* (2nd ed.). Redwood City, CA: Benjamin/Cummings.
- American Association for Artificial Intelligence. (1999). *AAAI fall symposium on question answering systems* (Tech. Rep. FS-99-02). Menlo Park, CA: AAAI Press.
- Beck, H. W., Mobini A. M., & Kadambari, V. A. Word is worth 1000 pictures: Natural language access to digital libraries. Retrieved June 15, 2004, from <http://archive.ncsa.uiuc.edu/SDG/IT94/Proceedings/Searching/beck/beckmain.html>
- Galitsky, B. (2003). Natural language question answering system: Technique of semantic headers. *Advanced Knowledge International*, Adelaide, Australia. Retrieved from <http://www.dcs.bbk.ac.uk/~galitsky/NL/book/>
- Gayatri, T. R., & Raman, S. (2001). Natural language interface to video database. *Natural Language Engineering*, 7(1), 1-27.
- Gazdar, G. (1999). Natural language interfaces to databases. Retrieved from <http://www.cogs.susx.ac.uk/lab/nlp/gazdar/teach/nlp/nlpnode157.html>
- Maybury, M. T. (2000). Adaptive multimedia information access—Ask questions, get answers. *First International Conference on Adaptive Hypertext (AH '00)*, Trento, Italy.
- Maybury, M. T. (Ed.) (2004). *New directions in question answering*. Cambridge, MA: MIT Press.
- Microsoft. English query. Retrieved June 15, 2004, from <http://www.microsoft.com/sql/evaluation/features/english.asp>
- Ng, H. T., Lai Pheng Kwan, J., & Xia, Y. (2001). Question answering using a large text database: A machine learning approach. *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP 2001)*, Pittsburgh, PA, June 3-4.
- Pasca, M. (2003). *Open-domain question answering from large text collections, center for the study of language and information*. Lecture notes.
- Popescu, A.-M., Etzioni, O., & Kautz, H. (2003). Towards a theory of natural language interfaces to databases. *Intelligent User Interfaces*.
- Sabourin, C. F. (1994). Natural language interfaces: Bibliography. Interfaces to databases, to expert systems, to robots, to operating systems, and to question-answering systems. *Infolingua*, Montreal, Canada.
- Wallace, M. (1984). *Communicating with databases in natural language*. Chichester, UK: Ellis Horwood.

## KEY TERMS

**Keyword Search:** A search for documents containing one or more words that are specified by a user.

**Natural Language Search:** Search in which one can ask a question in natural English (such as, Where can I find information on William Shakespeare?) as opposed to formulating a search statement (such as, su:Shakespeare, William).

**Natural Language Understanding:** A problem of conversion of a natural language expression into its formal representation.

**Phrasing:** The manner in which something is expressed in words.

**Pragmatics:** The study of the contribution of contextual factors to the meaning of what language users say.

**Semantics:** The branch of linguistics that studies meaning in language. One can distinguish between the study of the meanings of words (lexical semantics) and the study of how the meanings of larger constituents come about (structural semantics). In the study of language, semantics is concerned with the meaning of words, expressions, and sentences, often in relation to reference and truth. Metasemantic theories study key semantic notions such as meaning and truth and how these notions are related.

**Syntax:** The grammatical arrangement of words in sentences.



# Normalizing Multimedia Databases

**Shi Kuo Chang**

*University of Pittsburgh, USA*

**Vincenzo Deufemia**

*Università di Salerno, Italy*

**Giuseppe Polese**

*Università di Salerno, Italy*

## INTRODUCTION

Multimedia databases have been used in many application fields. As opposed to traditional alphanumeric databases, they need enhanced data models and DBMSs to enable the modeling and management of complex data types. After an initial anarchy, multimedia DBMSs (MMDBMS) have been classified based on standard issues, such as the supported data model, the indexing techniques to support content-based retrieval, the query language, the support for distributed multimedia information management, and the flexibility of their architecture (Narasimhalu, 1996).

A conspicuous number of MMDBMS products have been developed. Examples include CORE (Wu, Mehtre, Lam, & Gao, 1995), OVID (Oomoto & Tanaka, 1993), VODAK (Löhr & Rakow, 1995), QBIC (Flickner et al., 1995), ATLAS (Sacks-Davis, Ramamohanarao, Thom, & Zobel, 1995), each providing enhanced support for one or more media domains among text, sound, image, and video. Some of these products support specific data models, whereas others support the object-oriented data model or even the canonical relational data model. Moreover, extensible relational DBMSs have been introduced to extend relational DBMSs with object-oriented features, such as the capability to manage complex data types, including multimedia data. In particular, they implement the concept of the object-relational universal server, providing means to enable the construction of user defined data types (UDTs), and functions for manipulating them (UDFs). In addition, SQL3 has become the standard for relational DBMSs extended with object-oriented capabilities. The standard includes UDTs, UDFs, LOBs (a variant of BLOBs), and type checking on user-defined data types, which are accessed through SQL statements. Early examples of extensible RDBMSs include Postgres, IBM/DB2 version 5, Informix, and ORACLE 8.

As MMDBMSs technology has become more mature, the research community has been seeking new method-

ologies for multimedia software engineering. Independently from the data model underlying the chosen MMDBMS, multimedia software engineering methodologies should include techniques for database design, embedding guidelines and normal forms to prevent anomalies that might arise while manipulating multimedia data.

In this paper, we describe a general-purpose framework to define normal forms in multimedia databases. The framework applies in a seamless way to images as well as to all the other different media types. The semantics of multimedia attributes is defined by means of generalized icons (Chang, 1996), previously used to model multimedia languages in a visual language fashion. In particular, generalized icons are used here to derive extended functional dependencies, which are parameterised upon the similarity measure used to compare multimedia data (Santini & Jain, 1999). Based on these new dependencies, we define three normal forms aiming to reach a suitable partitioning of multimedia data and to derive database schemes that prevent possible manipulation anomalies.

## BACKGROUND

The normalization of multimedia databases needs to account for many new issues as opposed to alphanumeric databases. Many different types of complex data need to be analysed. However, in the literature we find many database design techniques focusing on image databases. In particular, a technique for normalizing image databases focuses on the partitioning of images so as to enhance image search and retrieval (Santini & Gupta, 2002). To this end, the technique aims to define dependencies among image features, which suggest to the designer how to efficiently map them into a database schema.

Database designers use their understanding of the semantics of attributes to specify functional dependencies among them (Elmasri & Navathe, 2003). As we know, other than alphanumeric data, multimedia databases can

store complex data types, such as text, sound, image, and video, which were initially modelled as binary large objects (BLOBs) in early MMDBMSs.

In the following we introduce a framework to model the semantics of multimedia attributes aiming to derive functional dependencies between them. To this end, we have exploited the framework of generalized icons (Chang, 1996). Generalized icons are dual objects  $(x_m, x_i)$ , with a logical part  $x_m$ , and a physical part  $x_i$ . They can be used to describe multimedia objects such as images, sounds, texts, motions, and videos. A generalized icon for modeling images is like a traditional icon, whereas those for modeling sounds, texts, motions, and videos are earcons, ticons, micons, and vicons, respectively. For all of them we denote the logical part with  $x_m$ , whereas the physical part will be denoted with  $x_i$  for icons,  $x_e$  for earcons,  $x_t$  for ticons,  $x_s$  for micons, and  $x_v$  for vicons. The logical part  $x_m$  always describes semantics, whereas  $x_i$  represents an image,  $x_e$  a sound,  $x_t$  a text,  $x_s$  a motion, and  $x_v$  a video. Furthermore, a multicon is a generalized icon representing composite multimedia objects (Arndt, Cafiero, & Guercio, 1997). Generalized icons can be combined by means of special icon operators. The latter are dual objects themselves, wherein the logical part is used to combine the logical parts  $x_m$  of the operand icons, whereas the physical part is used to combine their physical parts  $x_i$ . For instance, by applying a temporal operator to several icons and an earcon, we might obtain a vicon, with the physical part representing a video, and the logical part describing the video semantics.

In our framework we associate a generalized icon to each complex attribute, using the logical part to describe its semantics and the physical part to describe the physical appearance based on a given storage strategy.

The logical parts of generalized icons will have to be expressed through a semantic model. Conceptual graphs are an example of a semantic model that can be used to describe logical parts of generalized icons (Chang, 1996). Alternatively, the designer can use frames, semantic networks, or visual CD forms (Chang, Polese, Orefice, & Tucci, 1994). As an example, choosing a frame-based representation, an image icon representing the face of a person may be described by a frame with attributes describing the name of the person, the colors of the picture, objects appearing in it, including their spatial relationships. A vicon will contain semantic attributes describing the images of the video photographs, the title of the video, the topic, the duration, the temporal relationships, and so forth.

Based on the specific domain of the multimedia database being constructed, the designer will have to specify the semantics of simple and complex attributes according to the chosen semantic model. Once he or she has accomplished this task, the generalized icons for the multimedia

database are completely specified, which provides a semantic specification of the tuples in the database.

As an example, to describe semantics in a database of singers we might use the attributes name, birth date, and genre as alphanumeric attributes; picture as an icon representing the singer's picture; one or more earcons to represent some of the singer's songs; and one or more vicons to represent the singer's video clips. A tuple in this database might describe information about a specific singer, including his or her songs and video clips. This provides a complete semantic specification of the tuple.

## NORMAL FORMS IN MULTIMEDIA DATABASES

In traditional relational databases, a functional dependency is defined as a constraint between two sets of attributes from the database. Given two sets of attributes  $X$  and  $Y$ , a functional dependency between them is denoted by  $X \rightarrow Y$ . The constraint says that, for any two tuples  $t1$  and  $t2$  having  $t1[X] = t2[X]$ , then  $t1[Y] = t2[Y]$ . This concept cannot be immediately applied to multimedia databases because there are no similar simple, efficient methods to compare multimedia attributes. In other words, we need a method for defining equalities between groups of attributes involving complex data types.

Generally speaking, the matching of complex attributes needs to be based on an approximate match paradigm, such as those used in content-based retrieval from multimedia databases (Schaubl, 1997). In particular, we extend the definition of functional dependency by selecting a specific similarity function and thresholds to perform approximate comparisons of complex data types. Thus, the functional dependencies change if we use different similarity functions. As a consequence, we enrich the notation used for functional dependencies to include symbols representing the chosen similarity function. In what follows, we introduce some basic concepts of similarity theory (Santini & Jain, 1999).

Tuples of a relation can be compared by means of a set of relevant features  $\Phi$ . For instance, images can be compared using attributes such as color, texture, and shape; audio data can be compared using loudness, pitch, brightness, bandwidth, and harmonicity. The values of each feature  $F \in \Phi$  belong to a domain  $D = \text{dom}(F)$ .

The similarity between two elements  $x$  and  $y$  in a tuple is based on distance measures in feature spaces (that are assumed to be metric spaces) or, equivalently, on similarity functions. In the following, we will always refer to distance functions, but it should be understood that the same considerations apply to similarity functions, given the symmetry between distance and similarity functions.

In particular, given two elements  $x$  and  $y$  belonging to a given data type  $X$ , we consider distance functions of type  $d: D^2 \rightarrow [0, 1]$ , such that for each  $v_x, v_y \in D$   $d(v_x, v_y)$  returns the distance of  $x$  from  $y$  with respect to the feature space  $D$ . In what follows we indicate such distance with  $d(x, y)$  for any two elements  $x, y: X$ . Moreover, given a data type  $X$ , we denote with  $D(X)$  the set of distance functions defined on  $X$ , and with  $X_d$  the feature space on which the distance function  $d$  is defined.

To evaluate the similarity between the multimedia objects of two different tuples, we introduce a *tuple distance function*, which summarizes the results produced by the different distance functions applied to the elements of the tuples. In particular, given a relation  $R(z_j:Z_1, \dots, z_n:Z_n)$ , if  $x = (x_1, \dots, x_n)$  and  $y = (y_1, \dots, y_n)$  are two tuples of  $R$ , then  $\varpi(x, y) = g(d_1(x_1, y_1), \dots, d_n(x_n, y_n))$  measures the distance between  $x$  and  $y$ , where  $d_i \in D(Z_i)$  and  $g: [0, 1]^n \rightarrow [0, 1]$  is a suitable monotonically nondecreasing function of its arguments. Notice that if  $n = 1$ , then  $\varpi(x, y) = d_1(x, y)$ . Given a set of data types  $X$ , we denote with  $TD(X)$  the set of tuple distance functions defined on  $X$ .

- **Definition 1:** Let  $R(z_j:Z_1, \dots, z_n:Z_n)$ ,  $\varpi$  be a tuple distance function on  $R$ ,  $t$  be a maximum distance threshold,  $x = (x_1, \dots, x_n)$  and  $y = (y_1, \dots, y_n)$  be two tuples in  $R$ , we say that  $x$  is similar within  $t$  to  $y$  with respect to  $\varpi$ , denoted with  $x \cong_{w(t)} y$ , if  $\varpi(x, y) \leq t$ .

Now we are ready to introduce the notion of functional dependency for multimedia databases, named *type-M functional dependency*.

- **Definition 2:** Let  $R$  be a relation with attribute set  $U$ , and  $X, Y \subseteq U$ .  $X_{g_1}(t') \rightarrow Y_{g_2}(t'')$  is a *type-M functional dependency* (MFD) relation if and only if for any two tuples  $t1$  and  $t2$  in  $R$  that have  $t1[X] \cong_{g_1(t')} t2[X]$ , then  $t1[Y] \cong_{g_2(t'')} t2[Y]$ , where  $g_1 \in TD(X)$  and  $g_2 \in TD(Y)$ , whereas  $t'$  and  $t'' \in [0, 1]$  are thresholds.

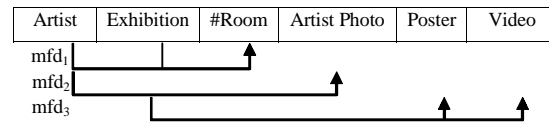
This means that the features used by  $g_2$  on  $Y$  depend on the features used by  $g_1$  on  $X$ ; or, alternatively, the values of the features used by  $g_1$  on  $X$  component imply the range of values for the features used by  $g_2$  on  $Y$  component.

As an example, if we define a functional dependency between attributes FINGERPRINT and PHOTO of a police database, and use the fingerprint matching function FINGERCODE for comparing digital fingerprints (Maltoni, Maio, Jain, & Prabhakar, 2003), and the similarity technique used by QBIC for comparing photo images, we would write as follows

$$\text{FINGERPRINT}_{\text{FINGERCODE}(t')} \rightarrow \text{PHOTO}_{\text{QBIC}(t'')}$$

This constraint says that for any two tuples  $t1$  and  $t2$  such that  $t1[\text{FINGERPRINT}]$  is considered similar within the threshold  $t'$  to  $t2[\text{FINGERPRINT}]$  by the FINGERCODE, then  $t1[\text{PHOTO}]$  is considered similar within the threshold  $t''$  to  $t2[\text{PHOTO}]$  by the QBIC. Given the semantics of the FINGERPRINT attribute, it is expected that the designer fix the value of  $t'$  to zero.

In traditional alphanumeric databases, normal forms are used to derive database schemes that prevent manipulation anomalies (Elmasri & Navathe, 2003). Similar anomalies can arise in a multimedia database. Thus, a multimedia database designer should take all the precautions at database design time to avoid such anomalies. As an example, let us consider the following relation of an art exhibition multimedia database schema, with some associated MFDs on its attributes:



It is easy to imagine how the schema of such a relation will yield manipulation anomalies, because it mixes together too many types of information. In fact, if we delete an exhibition from the database, we might lose an artist's data, such as his or her photo. Moreover, to perform such an operation we need to perform deletion across all the tuples referring to that exhibition, because the relation schema yields duplication of information regarding an exhibition. If we are willing to insert an artist's picture, without knowing the data of an exhibition, we have to leave some blank attributes, one of which belongs to the primary key. To avoid such anomalies, we should have stored information about exhibitions separately from those of artists.

Other types of anomalies arise from grouping too many multimedia data in the same multimedia attribute. For instance, the soundtrack of a video might be useful for different queries and multimedia presentations; hence, we need to store the video and its soundtrack as two separated multimedia attributes, together with synchronization information to coordinate their combination. Thus, we need to segment the video associated to the video. Synchronization information can be stored as metadata or as special relations, depending on the specific MMDBMS technology used.

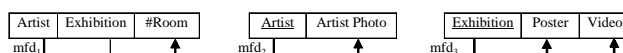
In this section we present three normal forms for multimedia databases. They are based on the type-M dependencies defined above. Therefore, their results depend on the distance functions used to derive dependencies and can be used to derive multimedia database schemes without potential manipulation anomalies. More-

over, the application of a normal form itself might need a specific manipulation function, such as a segmentation function, to handle a specific media type.

Our first normal form for multimedia attributes (1MNF) regards the granularity of multimedia attributes. We say that a multimedia database schema is in *first multimedia normal form* (1MNF) if each attribute  $A$  has the type of number, string or elementary generalized icon. For instance, image attributes can be decomposed in a certain number of  $k$  image components, which will be stored as separated attributes. In this case, the application of the normalization process is based on a specific segmentation function. This normalization process can also be applied to composite multimedia objects to decompose them into elementary generalized icons, which will be stored in separate attributes. For instance, we might have a vicon attribute and want to separate sounds from images. Again, the application of these normal forms requires the availability of specific segmentation function. Moreover, the decomposition of a composite multimedia attribute may require the storing of additional data structures to enable the reconstruction of the original attribute format. In particular, such data structure should store the relations between the different attribute components. As an example, if we store the different components of a segmented image separately then we need to store a data structure containing information such as the relative position of each image segment and possibly some spatial relation with respect to other segments. Such metadata is associated to the multimedia attribute, and the way it is stored depends on the specific type of DMBS used.

We say that a multimedia database schema is in *second multimedia normal form* (2MNF) if it is in 1MNF and each nonprime attribute  $A$  is fully dependent on the primary key. In case there is a partial dependency of  $A$  from a subset  $\{k_p, \dots, k_j\}$  of key attributes, then the designer can decide to normalize the schema by splitting the original schema  $R$  into two subschemes  $R_1 = R - T$  and  $R_2 = \{k_p, \dots, k_j\} \cup T$ , where  $T = \{A\} \cup \{B_i \mid B_i \in R, \{k_i, \dots, k_j\}_{s_1} \rightarrow \{B_i\}_{s_2}\}$ . For brevity, in the following we omit the threshold from the similarity expressions.

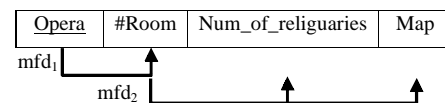
As an example, let us analyse the MFDs and the normal forms of the relation schema from the art exhibition multimedia database seen previously.  $\{\text{Artist, Exhibition}\} \rightarrow \#\text{Room}$  is a full dependency, but  $\{\text{Artist, Exhibition}\} \rightarrow \text{ArtistPhoto}_{d_1}$  and  $\{\text{Artist, Exhibition}\} \rightarrow \{\text{Poster, Video}\}_{d_2}$  are partial dependencies because of  $\text{mfd}_2$  and  $\text{mfd}_3$ , respectively. These MFDs lead to the decomposition of the relation into the following three relations, each of which is in 2MNF.



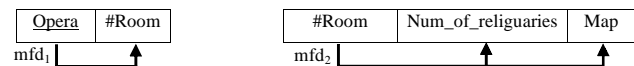
We say that a multimedia database schema is in *third multimedia normal form* (3MNF) if it is in 2MNF and the nonprime attributes are not mutually dependent. Equivalently, we can say that whenever a MFD  $X_{s_1} \rightarrow A_{s_2}$  holds in  $R$ , either

- a.  $X$  is a superkey of  $R$ , or
- b.  $A$  is a prime attribute of  $R$ .

As an example, let us consider the following simple multimedia relation:



The dependency  $\text{mfd}_2$  violates 3MNF because  $\#\text{Room}$  is not a super key of the relation, and  $\text{Num\_of\_Reliquaries}$  and  $\text{Map}$  are not prime attributes. We can normalize the relation schema by decomposing it into the following two 3MNF relation schemas. We construct the first relation by removing the attributes violating 3MNF, namely  $\text{Num\_of\_Reliquaries}$  and  $\text{Map}$ , from the original relation, and placing them with  $\#\text{Room}$  into the second relation.



## FUTURE TRENDS

Because the multimedia dependencies and multimedia normal forms depend on the tuple distance functions, by imposing additional constraints on tuple distance functions we can introduce more restricted multimedia dependencies and multimedia normal forms. This makes the proposed framework very flexible to accommodate the requirements of different applications. Furthermore, other normal forms can be introduced if we explore more deeply certain application domains, such as e-learning.

In the future, we aim to further explore the theoretical framework and, according to feedback resulting from its application, to more application fields.

## CONCLUSION

In this paper we have proposed a framework for the normalization for multimedia databases. Our goal here it has been to derive proper design guidelines to be adopted within multimedia software engineering methodologies. The framework is parametric with respect to the distance



functions used for the different media domains, and allows many degrees of normalization, depending on the distance function and the segmentation techniques used to decompose complex multimedia attributes.

## REFERENCES

Arndt, T., Cafiero, A., & Guercio, A. (1997). *Multimedia languages for teleaction objects*. *Proceedings of the IEEE Symposium on Visual Languages, VL97*, Capri Island, Italy.

Chang, S. K. (1996). Extending visual languages for multimedia. *IEEE Multimedia Magazine*, 3(3), 18-26.

Chang, S. K., Polese, G., Orefice, S., & Tucci, M. (1994). A methodology and interactive environment for iconic language design. *International Journal of Human Computer Studies*, 41, 683-716.

Elmasri, R., & Navathe, S. B. (2003). *Fundamentals of database systems* (4th Ed.). Reading, MA: Addison Wesley.

Flickner, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., et al. (1995). Query by image and video content: The QBIC system. *IEEE Computer*, 28(9), 23-32.

Löhr, M., & Rakow, T. C. (1995). Audio support for an object-oriented database-management system. *Multimedia Systems*, 3(6), 286-297.

Maltoni, D., Maio, D., Jain, A. K., & Prabhakar, S. (2003). *Handbook of fingerprint recognition*. New York: Springer Verlag

Narasimhalu, A. D. (1996). Multimedia databases. *Multimedia Systems*, 4(5), 226-249.

Oomoto, E., & Tanaka, K. (1993). OVID: Design and implementation of a video-object database system. *IEEE Transactions on Knowledge and Data Engineering*, 5(4), 629-643.

Sacks-Davis, R., Kent, A., Ramamohanarao, K., Thom, J., & Zobel, J. (1995). Atlas: A nested relational database system for text applications. *IEEE Transactions on Knowledge and Data Engineering*, 7(3), 454-470.

Santini, S., & Gupta, A. (2002). Principles of schema design for multimedia databases. *IEEE Transactions on Multimedia*, 4(2), 248-259.

Santini, S., & Jain, R. (1999). Similarity measures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9), 871-883.

Schaubl, P. (1997). *Multimedia Information retrieval: Content-based information retrieval from large text and audio databases*. Boston: Kluwer.

Wu, J. K., Narasimhalu, A. D., Mehtre, B. M., Lam, C. P. & Gao, Y. J. (1995). CORE: A content-based retrieval engine for multimedia information system. *Multimedia Systems*, 3(1), 25-41.

## KEY TERMS

**Distance Function:** A function used to compute the similarity between two multimedia elements. In particular, it returns a number in the range [0, 1], with values close to 0 for similar elements.

**Extensible RDBMS:** Relational DBMS implementing the concept of universal server. It allows the extension of the DBMS type system by enabling the definition of user defined data types (UDT), and the associated user defined manipulation functions (UDF).

**Generalized Icons:** Generalized icons are used to describe multimedia objects such as images, sounds, texts, motions, and videos. They are dual objects with a logical part and a physical part.

**MMDBMS:** DBMSs with enhanced functionalities to efficiently organize, store, and retrieve multimedia objects.

**Multimedia Normal Forms:** Rules for testing multimedia database schemes in order to prevent possible manipulation anomalies.

**Tuple Distance Function:** A function used to compute the similarity between two tuples of a relational multimedia database. Similarly to the distance function, this function also returns a number in the range [0, 1], computed by synthesizing the values that the distance function returns when applied on the attributes of the compared tuples.

**Type-M Functional Dependency:** Let R be a relation with attribute set U, and  $X, Y \subseteq U$ .  $X_{g_1}(t') \rightarrow Y_{g_2}(t'')$  is a *type-M functional dependency* (MFD) relation if and only if for any two tuples t1 and t2 in R that have  $t1[X] \cong_{g_1}(t')$   $t2[X]$ , then  $t1[Y] \cong_{g_2}(t'')$   $t2[Y]$ , where  $g_1 \in TD(X)$  and  $g_2 \in TD(Y)$ , whereas  $t'$  and  $t'' \in [0, 1]$  are thresholds.



# Object Modeling of RDBMS Based Applications

**Giuseppe Polese**

*Università di Salerno, Italy*

**Vincenzo Deufemia**

*Università di Salerno, Italy*

**Gennaro Costagliola**

*Università di Salerno, Italy*

**Genny Tortora**

*Università di Salerno, Italy*

## INTRODUCTION

As the development of industrial software applications become more complex, the benefits of developing a comprehensive “blueprint” enabling developers to visualize the scope of a project increase substantially.

One of the key points in the development of most industrial applications is the design of the database. The object-oriented design paradigm enhances database modeling because the object model is expressive, concise, and relatively easy to use. On the other hand, the relational data model offers the advantages of a standard model, having a rigorous theoretical foundation, a mature technology, and the possibility to easily work with declarative languages. The complementary strengths of object and relational models makes it desirable to have them coexist in some form, achieving the best from both their paradigms (Li & Zhao, 2003; Musto et al., 2000).

The Object Modeling of Relational Applications (OMAR) methodology uses object-oriented front end modeling and supports the generation of relational database schemes for the designed applications (Musto et al., 2000). It uses inference mechanisms exploiting the data structuring information and the dynamic models produced during the object-oriented modeling phases to infer an appropriate relational data schema and the associated manipulation mechanisms. Thus, a designer still benefits from the advantages offered by the object-oriented modeling paradigm, without losing the above mentioned advantages offered by the relational data model.

In this article, we show how we have used visual language technology to develop OMAR front-end CASE tools. A visual compiler generator, VLCC, has been used to model the syntax of UML diagrams and to generate a syntax driver editor and a compiler for them. Moreover, the inference mechanisms of OMAR have been embedded

within VLCC by means of semantic routines. These are invoked upon the successful parsing of UML diagrams and produce an object/relational source code version of the application under development. We describe such a version of the code by using the PC++ language (Musto et al., 2000), a high level C++ providing abstract mechanisms to manage the communication with RDBMSs.

## BACKGROUND

Applications that combine an object-oriented development method with RDBMS-based implementation can benefit from the use of a powerful, flexible, and expressive design paradigm and a standard implementation platform. In this way, application experts can focus on design instead of low-level tasks like tuning a database, just as high-level languages free programmers from repetitive tasks like assigning variables to registers. Any architecture that preserves this separation of concerns must have the ability to hide low-level details from the application expert. One way to do this is to automate mapping rules through an object-model compiler, having an RDBMS as its target architecture. Experts can then focus on the nuances of the application and avoid rediscovering database design principles for each new project. Moreover, the accuracy of the tables no longer depends on the person doing the conversion. One of the most important approaches for mapping object schemes onto RDBMS schemes is presented in Blaha, Premerlani, and Shen (1991). This approach combines the Object Modeling Tool (OMT) with Schemer, a compiler developed at GE Corporate R&D. In particular, Schemer converts the object schemes into SQL code, which can then be used to generate relational tables.

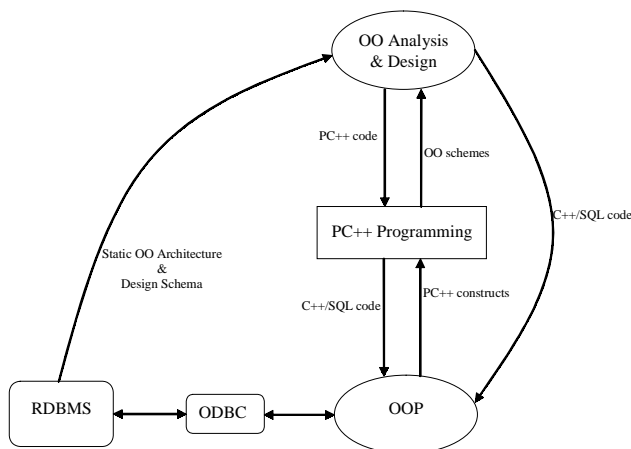
## THE DEVELOPMENT PROCESS WITH OMAR

Nowadays, there is still high demand of compatibility between object-oriented development methodologies and traditional DBMS platforms. In fact, in spite of reduced modeling capabilities, traditional RDBMSs offer the advantages of a widely used technology that also allows to easily recruit or train experienced professionals in the field. On the other hand, the object-oriented paradigm has the potential to improve many aspects of the software development process, including a more effective maintenance and reuse of existing applications.

The OMAR methodology supports the development of RDBMS-based applications starting from object-oriented specifications or programs. It allows a *designer* to build large *systems* by *using* object-oriented design methodologies and to generate a prototype of the whole relational application with the support of automated tools. The OMAR development life cycle is depicted in Figure 1.

The PC++ programming environment relies upon the PC++ compiler, which extends the C++ language with abstract mechanisms to cope with persistent classes and objects to be stored and manipulated through relational DBMSs. The programmer uses high-level constructs to declare persistent classes and the object-oriented paradigm to interact with the stored data. It will be the compiler responsibility to generate the appropriate C++/SQL code using ODBC calls (Gryphon et al., 2000). The PC++ programming environment can be used as an entry point of the OMAR methodology for developing small relational applications, or it can be used as a back end for large applications modeled through the object-oriented design paradigm of OMAR. The object-oriented design schemes can be translated either in C++/SQL or PC++ code. The

Figure 1. The OMAR development life cycle



latter can in turn be translated in C++/SQL using the PC++ compiler.

## MAPPING CLASS DIAGRAMS ONTO RELATIONAL APPLICATIONS

In the following, we provide algorithms to capture and process the information contained within class diagrams, state diagrams, and other behavior diagrams of an object design schema to derive an appropriate structuring of a relational data schema for the application under development and the associated manipulation code. In particular, the class diagram provides the necessary information to derive the tables to be generated. The remaining diagrams can be exploited to analyze the dynamic behavior of the system, which provides heuristics on how to refine the relational data schema to accommodate the data navigation requirements of transactions, and it is used to derive the manipulation code.

### Extracting a Relational Data Schema from the Class Diagram

The rules for extracting a relational data schema from the class diagram of an application are an extension of the rules for mapping Extended Entity Relationship diagrams onto relational data schemes (Elmasri & Navathe, 2003). A sample of these rules is the mapping of Generalization/Specialization hierarchies.

A generalization/specialization hierarchy can be of different types depending upon two properties: disjunction/overlapping and partial/total (Elmasri & Navathe, 2003). Their combination yields the following four types of hierarchies: (1) disjoint/total, (2) disjoint/partial, (3) overlapping/total, and (4) overlapping/partial. If a generalization/specialization hierarchy of type 2, 3, or 4 has superclass  $C$  with attributes  $\{a_1, a_2, \dots, a_n\}$  and  $m$  subclasses  $\{S_1, S_2, \dots, S_m\}$ , then it is mapped by using the following rules:

- Create a table  $T_c$  with columns  $\{a_1, \dots, a_n\} \cup \{OID_c\}$  corresponding to the attributes of class  $C$  plus its unique identifier (OID). The latter is used to manage inter-object references and will be a candidate primary key for the table  $T_c$ .
- For each subclass  $S_i$ ,  $0 < i < m + 1$ , create a table  $T_i$  with columns  $\{attributes\ of\ S_i\} \cup \{OID_{S_i}\}$ . Once again, here the OID will be a candidate primary key for the table  $T_i$ .
- Create a system table INHERITANCE, if it does not exist. This table will be updated with the insertion of

## Object Modeling of RDBMS Based Applications

a new record (*superclass-name*, *subclass-name*) for each subclass  $S_i$ ,  $0 < i < m + 1$  and will be used for reverse engineering purposes.

As for generalization/specialization hierarchies of type 1, these are mapped by generating relational tables only for the subclasses  $\{S_1, S_2, \dots, S_m\}$  with columns  $\{attributes\ of\ S_i\} \cup \{OIDs_i\} \cup \{a_1, \dots, a_n\}$  corresponding to the attributes of the subclasses, plus their OIDs, plus the attributes of the superclass.

### Refining the Application through the Dynamic Constructs of the Object Model

So far we have provided the mapping of static constructs of the object model (constructs of the class diagram) on a relational data schema. The latter needs to be refined in order to satisfy the data navigation requirements of the transactions to be implemented in the application under development. To do this, we use an inference engine that examines the dynamic models developed during the Object Analysis and Design (OOAD) phases of OMAR to infer the characteristics of the transactions. This has the two-fold goal to enhance the relational data schema and to generate the structure of the code implementing the transactions. In particular, our engine exploits the functional models and the State Diagrams. The functional models highlight the dependencies between certain data through the functions contained in the Data Flow Diagrams. In fact, we can examine these diagrams to verify whether the functions use intermediate data to relate a pair of data. Thus, we can use Data Flow Diagrams to infer some functional dependencies to be used for normalization purposes. The data flows of a Data Flow Diagram correspond to attribute values of a Class Diagram, whereas the processes correspond to actions or activities of the State Diagram for a class.

The State Diagram for a class shows the dynamic behavior of the class. The examination of the State Diagrams can help detect classes that are not significant enough to have a table associated in the underlying relational database schema. For instance, this happens when a class is associated to a State Diagram with states that do not have a Data Flow representation, meaning that they only perform elementary operations. Thus, if the class had previously been associated to a table, the inference engine tries to eliminate such a table. To do this, it examines the complexity of the class structure in the Class Diagram and verifies whether the name of the class appears as a Data Store item in one of the Data Flow diagrams associated to a related class.

In the following, we show the Prolog code for this portion of the inference engine:

```
Makepersistent(CLASSNAME)
  ←
  class(CLASSNAME, LIST OF ATTRIBUTES),
  DataStore(CLASSNAME).
```

It makes the class CLASSNAME persistent if it is indicated as transient, but it appears as a data store in a Data Flow Diagram.

```
SuperPrimaryKey(LIST OF ATTRIBUTES, TABLENAME)
  ←
  Function(FUNCTIONNAME, LIST OF
  ATTRIBUTES, TABLENAMEID), ExistsTable(TABLENAME).
```

LIST OF ATTRIBUTES is a Primary Key for the table TABLENAME if there exists a function capable of selecting the class instances to be stored in the table TABLENAME by using the attributes in LIST OF ATTRIBUTES.

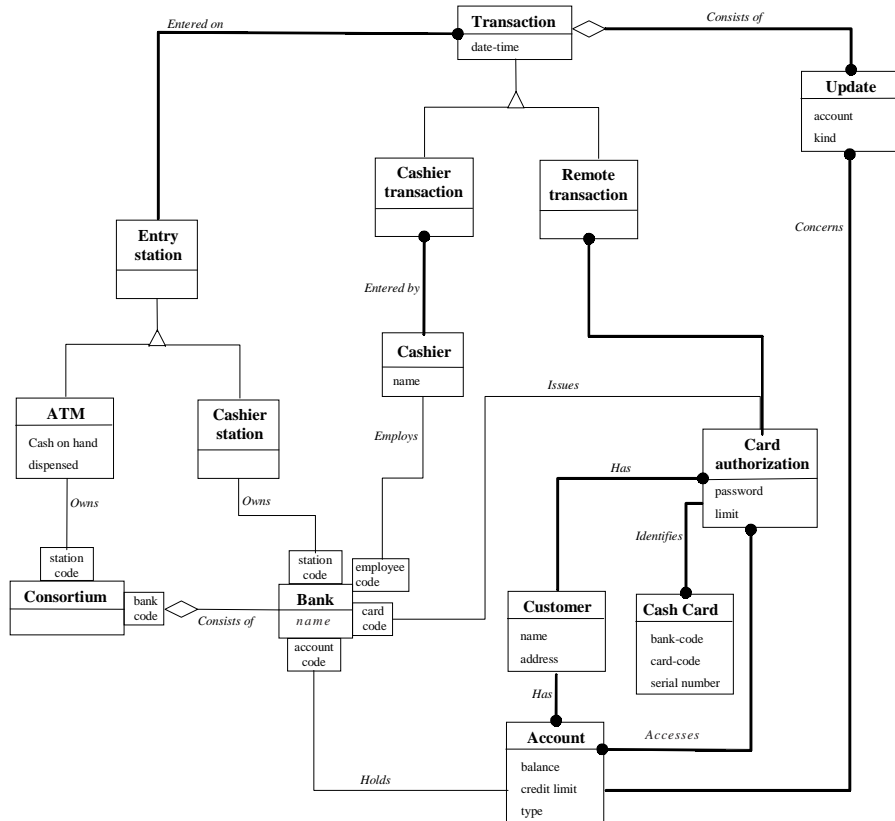
### Mapping the Object Schema of a Bank ATM Network

The OMAR methodology has been developed and experimented based upon the models used in the OMT methodology (Blaha et al., 1991) and later extended to the UML languages. We describe the application of the OMAR methodology for the example given in Blaha et al. (1991). It regards the development of a software system to support a computerized banking network including both human cashiers and automatic teller machines (ATMs) to be shared by a consortium of banks. The class diagram developed during the object-oriented analysis of this problem is given in Figure 2.

In this example, the classes that are indicated as persistent are: *Card-Authorization*, *Cash-Card*, *Customer*, *Account*, and *Bank*. Parts of the State Diagrams for the classes *ATM*, *Consortium*, and *Bank* are shown in Figures 3(a), 3(b), and 4(a). A Sequence Diagram for the Bank ATM networks is shown in Figure 4(b). A portion of the relational data schema produced by the application of the mapping rules on the ATM class diagram is shown in Figure 5.

At this point, the inference engine uses the dynamic models to refine the schema. For instance, from the State Diagrams of the classes *ATM*, *Consortium*, and *Bank*, given in Figures 3(a), 3(b), and 4(a), the engine detects the message exchanges between *ATM* and *Bank* going through *Consortium*. The latter has elementary actions only, that is, actions with no Data Flow Diagram associated. Moreover, from what said above, this class has been translated into a table because it is transient, but it

Figure 2. Class diagram for the bank ATM network



is related to the persistent class *Bank*. Thus, the engine tries to eliminate this table, also because the associated class *Consortium* has a reduced structure. The sequence diagram reveals the functional dependence of all the attributes of the class *Cash Card* on the combination of attributes *bank-code* and *card-code*. Thus, the pair (*bank-code*, *card-code*) can be a Primary Key for the table associated to the class *Cash Card*.

### OMAR PROGRAMMING WITH PC++

The transition from modeling to programming in OMAR is seamless. In other words, in both cases, the developer uses the object-oriented paradigm and exploits automated support for the generation of the final relational application.

Figure 3. Sample of the state diagram for class *ATM* (a) and a sample of the state diagram for class *Consortium* (b)

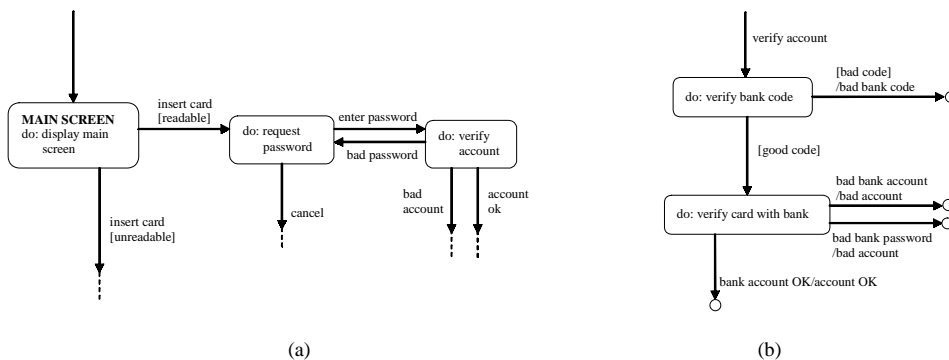


Figure 4. Sample of the state diagram for class bank (a) and a sequence diagram for bank ATM network (b)

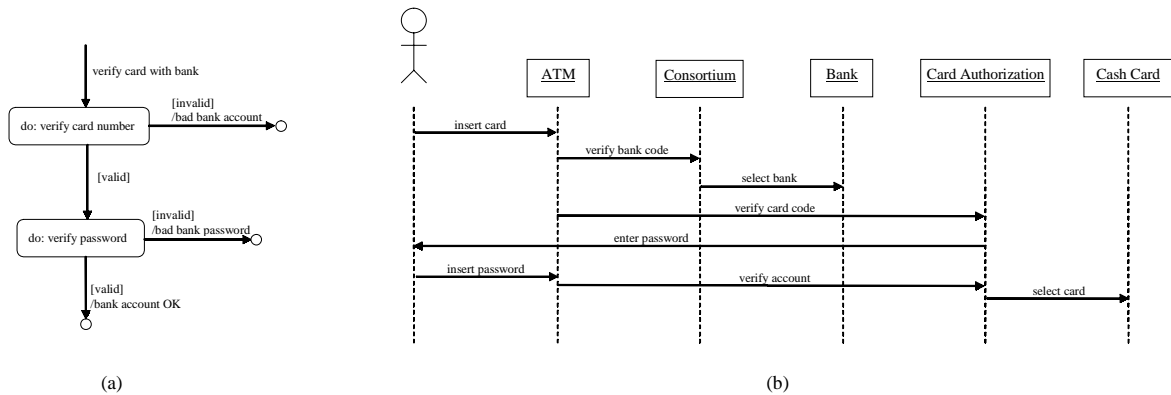


Figure 5. Some of the relational tables derived from the class diagram for the bank ATM network

BANK	
Attribute-name	Domain
bankID	ID
name	alphanumeric
bankcode	alphanumeric

Primary key: bankID

REMOTE-TRANSACTION	
Attribute-name	Domain
remote-transactionID	ID
date-time	date
card-authorizationID	ID

Primary key: remote-transactionID  
Foreign key: card-authorizationID

CASHIER	
Attribute-name	Domain
cashierID	ID
name	alphanumeric
employeeecode	alphanumeric

Primary key: cashierID

EMPLOYEES	
Attribute-name	Domain
cashierID	ID
bankID	ID
employeeecode	alphanumeric

Primary key: (bankID,employeeecode)  
Foreign keys: bankID, cashierID, employeeecode

PC++ is a C++ with added mechanisms for the management of object persistency onto relational DBMSs. Programming relational applications with PC++ is made simpler because the programmer can model persistent classes directly by using the DBClass declaration provided by PC++. The latter will be responsible for compiling these constructs, generating the equivalent C++ code, including the low level SQL instructions to handle the communication with the RDBMS. In particular, PC++ generates modules for communicating with an ODBC server.

Similar environments are *POET* (Poet Software, 1993) and *Persistence* (Persistence Software, 1993). As opposed to PC++, POET is not open to different RDBMS platforms, whereas Persistence realizes open RDBMS connectivity by using different techniques like the manipulation of the class constructors.

## Persistency in PC++

PC++ uses the technique of data manipulation language (DML) binding to provide type-orthogonal persistency (Bertino, 1991). The PC++ programmers can choose the classes and the instances that they want to make persistent. They use a new key word, namely, *dbclass* to declare persistent classes. PC++ accomplishes the translation from the object to the relational model by extending persistent classes with a new data member, *OID*, and with new methods to handle the communication with the RDBMS through ODBC, for example, a new method *Store()* to enable the storage of class instances. Every class is extended to provide persistency also for the instances of classes related to a persistent class, for instance, through a part-of or inheritance relation.



## Mapping Object Schemes onto PC++ Programs

Some rules for translating the static diagrams of the object schema are given below. Also here, the inference engine refines the products of translation by using the information contained within the dynamic models of the object-oriented schema.

1. For each transient class, generate a PC++ class with same name, attributes, and methods. In this case, the syntax is equivalent to that of a C++ class.
2. For each persistent class, generate a PC++ dbclass with same name, attributes, and methods.
3. For each generalization/specialization relation, insert the name of the superclass in the inheritance chain of the subclass.
4. For each binary bidirectional relation, insert a macro *Association(class-name, multiplicity)* within the private methods of the classes involved in the relation. Class-name is the name of the other class involved in the relation, whereas multiplicity is the cardinality of the relation. The PC++ compiler will use the *friend* mechanism to prevent update anomalies.
5. For each qualified bidirectional association relation, insert a macro *Association(class-name, multiplicity, association-name)* within the private methods of the classes involved in the relation. In addition to normal relations, here there is the additional argument *association-name* indicating the name of the relation. Moreover, an additional macro *Make-Class(relation-name, identifier<sub>1</sub>, identifier<sub>2</sub>, qualifier)* is generated where *relation-name* is the name of the qualified relation, *identifier<sub>1</sub>* and *identifier<sub>2</sub>* are the instance identifiers of the two classes involved in the relation, and *qualifier* is the qualifier attribute. The macro *Make-Class* causes the PC++ compiler to create the table *relation-name* with fields *identifier<sub>1</sub>*, *identifier<sub>2</sub>*, *qualifier*, and the pair (*identifier<sub>1</sub>*, *qualifier*) as the primary key. The attribute *qualifier* is also inserted as a data member of the qualified class.

## USING VISUAL LANGUAGE GENERATORS TO CONSTRUCT OMAR FRONT-END CASE TOOLS

VLCC is a grammar-based system that can support the implementation of graphical objects, syntax, and semantics of many types of visual languages (Costagliola et al.,

1997). VLCC uses the extended positional grammar model as its underlying grammar formalism (Costagliola & Polese, 2000). In this section, we show how the VLCC system has been used for generating OMAR front-end CASE tools.

The VLCC system architecture includes two main modules: the Grammar Editor and the Visual Programming Environment Generator. The first is a parametric system that can be configured to create and manipulate graphical components in a given visual language class and provides the language designer with editors to define specific visual grammars. The latter inputs the grammar specifications produced by Grammar Editor and generates an integrated visual programming environment containing a compiler and a visual editor for the defined visual language.

We used VLCC for the automatic generation of the visual programming environment for the UML visual notations. We began constructing a grammar for the Class Specification language (Booch, Jacobson & Rumbaugh, 1999; OMG, 2003). The Class Specification language describes the objects, the classes, the types, and the relationships that are in a class diagram. In this language, conceptual aspects and software interfaces are considered, whereas implementation aspects are considered into another part of the UML language, the Implementation Class Diagram Language. Class Specification is composed of "Class Identification" and "Class Diagram Specification". Class Identification specifies any system class and any system type. Class Diagram Specification conceptually specifies the existing relationships between classes and the class classifications. Therefore, we have implemented the grammar for Class Specification that includes three grammars, one for Class Identification, one for Class Diagram Specification, and one comprehending the preceding grammars.

The grammar formalism underlying VLCC allows a more flexible utilization of UML, since it gives the possibility to customize the visual notation and the rules for mapping the UML diagrams to lower level models.

The semantic routines to generate PC++ code from UML diagrams are inserted within YACC-like files (Johnson, 1978) generated by VLCC. An example of semantic routines included within the Class Specification language is the following:

```
classg →.....
→CLA
{ * syntactic attributes valuation *
  if (flag_list==0){
list=MakeList();
node=MakeNode("","");
Persistent(node, $1→GetTxtDirect(1));
Mlist=ParseMemberString($1→GetTxtDirect(2), out);
Node=MemberListInsert(node, Mlist);
List=TailInsert(list, node);
```

## Object Modeling of RDBMS Based Applications

```
flag_list=1;
}
else
{node=SearchClass(ClassName($1!GetTxtDirect(1)), list);
  if (node==NULL){
    node=MakeNode("", "");
    Persistent(node, $1→GetTxtDirect(1));
    Mlist=ParseMemberString($1→GetTxtDirect(2), out);
    Node=MemberListInsert(node, Mlist);
    List=TailInsert(list, node);
  }
};
```

With this semantic routine, the C++ SearchClass (ClassName(\$1→GetTxtDirect(1)), list) function checks if it already exists a node for *class-name* class. If it does not exist, then it is created by the C++ MakeNode() function. This list node contains information about class name, the persistency, and class members. This check is necessary to avoid that the same information is stored more times because of the tokens reinstate technique in visual grammars.

If a designer provides quantitative data together with diagrams, the generated CASE tools provide automatic support for workload computation and performance analysis of the modeled database. It is also capable of generating access tables, which it uses to contrast alternative design schemes and to choose the one guaranteeing best performances.

## FUTURE TRENDS

Several issues deserve further investigation. The OMAR inference engine is being extended to include fuzzy mapping rules. In particular, we are defining criteria for better refining the relational data scheme.

Moreover, we plan to extend the OMAR methodology to generate extended RDBMS schemes for multimedia applications (Polese, Tortora & Pannella, 2000). In particular, we use to construct customized versions of UML diagrams to model multimedia classes. Successively, these classes can be better described by using specific multimedia models such as Petri Nets.

## CONCLUSION

We have presented the methodology OMAR to design applications with the object-oriented paradigm and to implement them on a relational DBMS through the support of automatic tools. We have used visual language technology to develop prototype front-end CASE tools supporting the OMAR modeling paradigm.

The combination of the object and relational paradigms allows companies to perform an easier migration to the object-oriented design and programming paradigms and to make them coexist with the standard reliable relational paradigm.

The visual language technology is used to develop prototype front-end CASE tools supporting the OMAR modeling paradigm. In particular, VLCC is used as a compiler for the graphical notations of the UML language. The inference mechanisms of OMAR are embedded within the generated CASE tools by means of YACC semantic routines.

## REFERENCES

- Bertino, E. (1991, October-December). Object-oriented database management systems: Concepts and issues. *IEEE Computer*, 24(4), 33-47.
- Blaha, M., Eddy, F., Lorensen, W., Premerlani, W., & Rumbaugh, J. (1991). *Object-oriented modeling and design*. Prentice Hall International.
- Blaha, M., Premerlani, W., & Shen, H. (1994). Converting OO models into RDBMS schema. *IEEE Software*, 11(3), (pp. 28-39).
- Booch, G., Jacobson, I., & Rumbaugh, J. (1999). *The unified modeling language reference manual*. Addison-Wesley.
- Costagliola, G., De Lucia, A., Orefice, S., & Tortora, G. (1997). A parsing methodology for the implementation of visual systems. *IEEE Transactions on Software Engineering*, 23(12), 777-799.
- Costagliola, G., Deufemia, V., & Polese, G. (2004). A framework for modeling and implementing visual notations with applications to software engineering. *ACM Transactions on Software Engineering and Methodology*, 13(4), 431-487.
- Elmasri, R., & Navathe, S.B. (2003). *Fundamentals of database systems* (4th ed.). Addison-Wesley.
- Gryphon, R., Charpentier, L., Oelschlaiger, J., Shoemaker, A., Cross, J., & Lilley, A.W. (2000). *Using ODBC: Special edition*. Addison-Wesley.
- Johnson, S.C. (1978). *YACC: Yet another compiler compiler*. Murray Hills, NJ: Bell Laboratories.
- Li, L., & Zhao, X. (2003). UML specification of relational database. *Journal of Object Technology*, 2(5), 87-100.

Musto, A., Polese, G., Costagliola, G., & Tortora, G. (2000). Automatic generation of RDBMS based applications from object oriented design schemes. *Proceedings of SAC 2000*, (pp. 398-402).

OMG (Object Management Group). (2003). *UML specification version 1.5*. Retrieved February 3, 2005, from <http://www.omg.org/technology/documents/formal/uml.htm>

Persistence Software. (1993). *Persistence technical overview*. San Marco, CA: Author.

Poet Software. (1993). *POET programmer's and reference guide*. Hamburg, Germany: Author.

Polese, G., Tortora, G., & Pannella, A. (2000). An extended relational data model for multimedia databases. *Journal of Visual Languages and Computing*, 11(6), 663-685.

## KEY TERMS

**Access Table:** A table listing the transactions to be implemented with an application. For each transaction, it shows the classes from the Cclass Ddiagram it needs to visit, including the number and types of accesses (Read, Write) in order to collect the data necessary for composing the final result.

**Extended RDBMS Schema:** A database schema specified according to the relational data model extended with new data types. RDBMSs allowing the extension of the basic Type System are those implementing the concept of Universal Server. Examples are *Oracle*, *Informix*, *DB2*, and *Illustra*.

**OID:** An object's identifier (OID) is an unchangeable value that uniquely identifies that object and distinguishes it from all other objects. It is separate from the state and invisible to the user. OIDs can be used to represent associations between objects.

**OMAR:** Object Modeling of Relational Applications. A software engineering methodology for the object modeling of applications to be implemented over an RDBMS.

**PC++:** Persistent C++ extends C++ language by providing persistence facilities.

**Visual Language:** Language characterized by a set of visual sentences, each composed of icons spatially arranged over a two-dimensional space.

**Visual Language Compiler:** A software tool capable of performing a two-dimensional parsing of an input visual sentence, deciding whether the sentence belongs to a given visual language. After parsing, it translates successfully parsed visual sentences into sentences of a target language, which can be visual or textual.

**VLCC:** Visual Language Compiler Compiler.

# Object–Relational Modeling in the UML

Jaroslav Zendulka

Brno University of Technology, Czech Republic

## INTRODUCTION

Modeling techniques play an important role in the development of database applications. One of the trends in current database management systems is that they become object-relational (Stonebraker & Brown, 1999). The most recent version of the SQL standard, SQL:1999, includes object-relational features, and a number of leading companies have already released packages that incorporate them.

Well-known modeling techniques for relational databases, such as entity-relational diagrams, do not support important features of object-relational databases. In addition, the development of a database application involves a close working relationship between the software and database developers. Software developers deal with object-oriented software development and use object-oriented modeling techniques, such as a logical class model, to represent the main view of the application, whereas database developers model, design, build, and optimize the database. The most successful projects are marked by a shared vision and clear communication of project details (IBM, 2001). A common modeling language and supporting development tools can provide good conditions for it.

The Unified Modeling Language (UML) was adopted as an Object Management Group (OMG) standard for object modeling in 1997. Since that time, it has become popular and widely used *and* provides several types of diagrams that visualize a system from different perspectives. From database-design point of view, a class diagram is the most important diagram. It shows a set of structural elements and their static relationships. This model can be used not only as documentation, but also for data definition language (DDL) statements generation. If we want to employ the UML as a modeling language for development of a database application where persistent data is stored in an object-relational database, it is necessary to add the ability to model features of these kinds of databases in an effective and intelligible way, and the UML provides a proper extension mechanism for it. Modeling of an object-relational database schema will be called *object-relational modeling* in this article.

If we accept three perspectives for drawing class diagrams (Fowler, 2003)—conceptual, specification, and implementation—the need for object-relational

modeling is only for the implementation, and possibly specification, levels. A conceptual model should be developed with little or no regard to implementation and a target database environment.

## BACKGROUND

The UML provides three extensibility mechanisms that make it possible to extend the language in controlled ways (Booch, Rumbaugh, & Jacobson, 1998; Object Management Group [OMG], 2003):

- stereotypes,
- tagged values, and
- constraints.

A *stereotype* extends a vocabulary of the UML. It allows the introduction of a new model element derived from one existing in the UML metamodel. A *tagged value* extends the properties of the UML's model elements. It is a keyword-value pair element that may be attached to any kind of a model element. The keyword is called a *tag*. A *constraint* extends the semantics of a model block by means of specifying conditions and propositions that must be maintained as true, otherwise the system described by the model is invalid. There are some standard stereotypes, tagged values, and constraints predefined in the UML. One of them is a stereotype <<Table>>, which is a stereotype of the UML class element.

The main purpose of the extendability mechanisms is to tailor the UML to the specific needs of a given application domain or target environment. It makes it possible to develop a predefined set of stereotypes, tagged values, and constraints, and notation icons that collectively specialize and tailor the UML for specific domain or process. Such a set is called a *profile*. Several profiles has already been accepted by OMG as standard profiles, but none of them is for data or object-relational modeling.

Several works that propose extensions of the UML for data and object-relational modeling has been presented. Most of the extensions have been proposed not for SQL:1999 but for Oracle8, because this database management system (DBMS) had provided object extensions



before SQL:1999 was published. Probably the most important proposal has been developed and implemented by Rational Software Corporation in their Rational Rose product, which is one of the best-known UML-oriented modeling tools. It provides support not only for object-oriented database modeling but also for relational database modeling (Rational Software Corp., 2001) and Oracle8 object-relational modeling. The Rose Oracle8 tool permits both forward and backward engineering of Oracle8 object-relational schemas.

Marcos, Vela, and Cavero (2001, 2003) proposed several stereotypes, tagged values, and constraints for the modeling of structured types, typed tables, ARRAY type, REF type, two types of methods in the SQL:1999, and for modeling of similar elements in Oracle8i.

The approach presented in this article is based on extensions for Oracle8 by Rational Software, but it can be used for SQL:1999, too. Examples are drawn in Rational Rose, with the Rose Oracle8 tool.

## OBJECT-RELATIONAL MODEL IN ORACLE8 AND SQL:1999

Both the SQL:1999 and the SQL dialect of Oracle8 (and more recent releases) extend the relational model in several important directions. Only those modeling features that are presented in this article are summarized. More on new features of the SQL:1999 can be found in Melton & Simon (2001) or in the standard specification (Database Language SQL, 1999). More information on the Oracle object-relational model is available in Oracle documentation (Oracle, 2003a, 2003b).

First, both Oracle8 and SQL:1999 relinquished the basic demand on the relation in the relational model—to be in the 1NF. The user can define *user-defined data types*. In Oracle, there are two types of user-defined data types: object types and collection types.

An *object data type* is an abstraction of a real-world entity, the representation of which will be stored in a database. An object type is a schema object with a name, set of *attributes* and *methods*. Each attribute is of either a built-in scalar data type or a user-defined data type. This allows for the defining object types with a complex data structure.

*Methods* of an object data type implement operations with the data type. Every object type has a system-defined constructor method.

An object data type is a template for *objects*. Objects can be instantiated by the constructor method of a given object type and stored in object tables. An *object table* can be viewed either as a single column table of row objects or as a multicolumn table. Each row object has

assigned a unique object identifier (OID). It can be system-generated or primary-key based. Oracle provides a built-in data type called REF to encapsulate references to row objects. This type can be used to implement links between objects.

There are two *collection types* available: array types and table types.

Both collection types are sets of data elements of the same type, but there are important differences between them. An *array type* (called VARRAY) is an ordered and bounded set, whereas a *table type* (called a *nested table*) is unordered and unbounded. In addition, a VARRAY data value is stored and retrieved as one data unit, whereas a nested table value is stored in a storage table with every element of the collection mapped into a row of the storage table.

Another object extension concerns views. Just as a relational view is a virtual table, an *object view* is a virtual object table. Using object views, it is possible to create virtual object tables with columns of both built-in and user-defined data types mapped to columns of relational or object base tables. Object views provide the ability to offer specialized or restricted access to data stored in relational and object tables. In addition, they provide the ability to view relational data as objects. The object view definition contains information about the object type of the view objects, the way of constructing OID, and the mapping SELECT statement.

The Oracle8 object-relational model can be described as a metamodel in the UML (Zendulka, 2001). Although most of Oracle8 extensions are included in SQL:1999, there are some differences. First, the terminology of SQL:1999 differs from that of Oracle8 (e.g., in SQL:1999, object data types are called *structured types* and object tables/views are referred to as *referenceable tables/views*). In addition, only array as a collection type is available in SQL:1999. On the other hand, Oracle8 does not support inheritance (more recent versions do) whereas SQL:1999 does.

## EXTENSION OF THE UML

The mapping used in Rational Rose Oracle8 can be perceived as a profile definition for object-relational modeling with Oracle8 as a target DBMS. The profile introduces several stereotypes, some constraints in a form of conventions, and tagged values. The tagged values have a form of schema-generation properties attached to a project, class, operation, and attribute. They contain such values as, for example, a WHERE clause of a view definition. Stereotypes of the profile are listed in Table 1.



## Object-Relational Modeling in the UML

Table 1. Stereotypes for the Oracle8 profile

Stereotype	UML model element
<<ObjectType>>	Class
<<ObjectView>>	Class
<<ObjectTable>>	Class
<<NestedTable>>	Class
<<RelationalTable>>	Class
<<RelationalView>>	Class
<<VARRAY>>	Class
<<ObjectView>>	Dependency

Relationships between schema elements are modeled as associations, aggregations, and dependencies. Dependencies are used primarily to express relationships between data types and structures that store values of these types.

A simple purchase order example shows how the basic object extensions of Oracle8 are modeled in the profile. A conceptual class diagram of the example is displayed in Figure 1. Only one operation (*sumLineItems()*) of the *Order* class is shown in the diagram. The *Order* and *Customer* classes contain a complex attribute of an *Address* type consisting of *street*, *city*, and *zip* attributes.

### Object Types

Object types are modeled by a stereotype <<ObjectType>>, which is a stereotype of a UML class element. Attributes of a given object type are modeled as attributes, methods as operations.

### Nested Object Types

An object type can contain an attribute of another object type. This is modeled as a unidirectional association from the outer object type to the nested object type. A role name of the nested type is the name of the corre-

Figure 1. A conceptual class diagram of a purchase order example

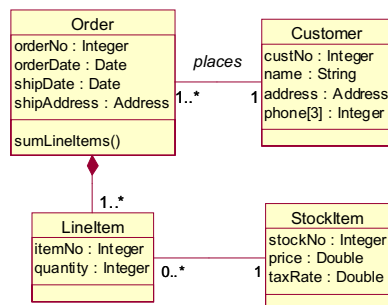
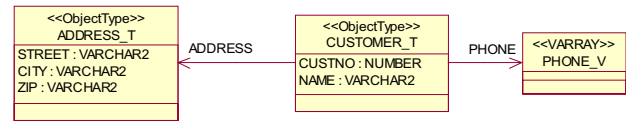


Figure 2. Modeling of a nested object type and an array



sponding attribute. Association in the profile implies a relationship *by value*.

### VARRAYs

A VARRAY collection type is modeled in the UML by a stereotype *VARRAY*, which is a stereotype of a UML class element. All elements of the array must be of the same type, either scalar or object one. If the array is based on an object type, the relationship is modeled as dependency of the VARRAY type on a given object type. A VARRAY can be used as an attribute of an object type. Association, as for nested object types, models this.

Assume that the *Customer* class from the example will be implemented by means of an object type *CUSTOMER\_T* with a nested type *ADDRESS\_T* and a *VARRAY* type attribute for a fixed-size array of phone numbers. The corresponding model in the UML is in Figure 2.

### Nested Tables

A nested table is modeled as a class with a stereotype <<NestedTable>>. If the nested table is based on an object type, the relationship is modeled as dependency of the nested table on a given object type, similarly as in VARRAY modeling. A nested table can be used as an attribute of an object type. Association, as for nested object types and VARRAYs, models this.

Let us assume that the *Order* class from our example will be implemented by means of an object type *ORDER\_T* with a nested type *ADDRESS\_T* and a nested table containing all items of a given order (of a

Figure 3. Modeling of a nested table

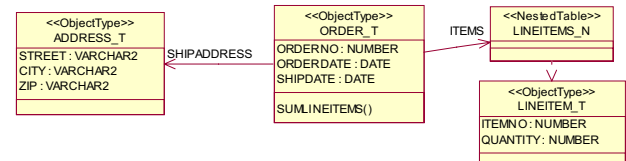
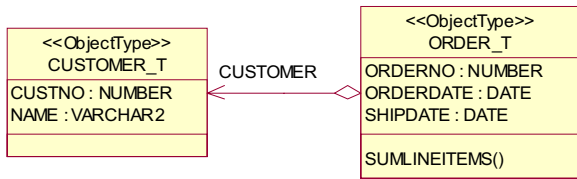


Figure 4. Modeling of a REF type



LINEITEM\_T type). The corresponding model in the UML is in Figure 3.

## REFs

A built-in Oracle8 REF type implements a reference to an object. The reference is modeled by aggregation. In the profile, aggregation implies “by reference” containment. Figure 4 shows that the *ORDER\_T* type contains an attribute *CUSTOMER* of the REF type, which references an object of the *CUSTOMER\_T* type.

## Object Tables

An object table is modeled as a class with a stereotype *<<ObjectTable>>*. Because object tables are built from underlying object types, this relationship is modeled as dependency. Figure 5 introduces an object table, *CUSTOMER*.

## Relational Tables

A relational table is modeled as a class with a stereotype *<<RelationalTable>>*. Attributes of the table that are of an object type, VARRAY, nested table, and REF are modeled in the same way as for object types.

## Object Views

An object view is modeled as a class with a stereotype *<<ObjectView>>*. Links between source tables (object or relational) and the object view are modeled as dependencies. The relationship with the underlying object type is modeled as a dependency with a stereotype *<<ObjectView>>*.

Figure 5. Modeling of an object table

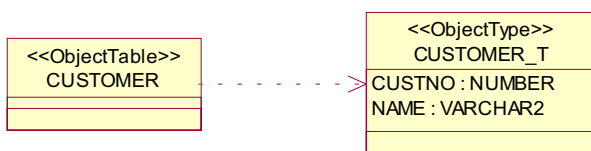


Figure 6. Modeling of an object view

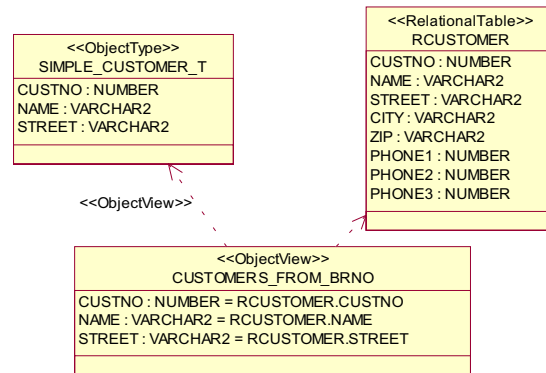


Figure 6 shows an example of an object view *CUSTOMERS\_FROM\_BRNO* with objects of a *SIMPLE\_CUSTOMER\_T* type. The view is defined on a relational table *RCUSTOMER*. The figure also shows how the columns of the source relational table are mapped on columns of the object type. It is one of conventions (constraints) defined in the profile. In addition, the condition from the WHERE clause of the view definition SELECT statement is stored in the *WhereClause* property (a tagged value) of the corresponding *<<ObjectView>>* class.

## Methods and Triggers

Methods of object types and database triggers are modeled as operations of classes with some conventions.

## FUTURE TRENDS

We can expect that object-relational modeling will become more common in the near future, because recent releases of DBMSs are more object-relational and object-oriented, and comply more with SQL:1999. This trend should also encourage the development of methods and tools for object-relational database design. Unfortunately, only a few such methods have been published (Mok & Paper, 2001). On the contrary, methods for a relational schema model generation exist and are implemented in computer-aided software engineering (CASE) tools (Rational Software Corp., 2001).

## CONCLUSION

The objective of this short paper was to introduce a UML profile for object-relational modeling. We focused on only fundamental features of the profile. Many

## Object-Relational Modeling in the UML

details, which are important for real projects, were omitted here (e.g., constraints and other model properties associated with the introduced elements). The profile can be used for modeling during both forward and reverse engineering.

## REFERENCES

Booch, G., Rumbaugh, J., & Jacobson, I. (1998). *The unified modelling language user guide*. Addison-Wesley Longman.

Database Language SQL—Part 2: Foundation. (1999). ISO/IEC 9075-2:1999 standard. *International Committee for Information Technology Standards*, Washington, DC.

Fowler, M. (2003). *UML distilled* (3rd Ed.). Addison-Wesley Longman.

IBM. (2001). Mapping objects to data models with the UML [White paper]. Somers, NY: IBM Corporation Software Group. Retrieved December 10, 2003, from <http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/tp185.pdf>

Marcos, E., Vela, B., & Cavero J. M (2001). Extending UML for object-relational database design, UML 2001. In M. Gogolla & C. Kobryn (Eds.), *LCNS 2185* (pp. 225-239). Heidelberg, Germany: Springer-Verlag.

Marcos, E., Vela, B., & Cavero, J. M. (2003). A methodological approach for object-relational database design using UML. *Software and Systems Modeling*, 1(2), 59-72.

Melton, J., & Simon, A. (2001). SQL: 1999. *Understanding relational language components*. Morgan Kaufmann.

Mok, W. Y., & Paper, D. P. (2001). On transformations from UML models to object-relational databases. *Proceedings of the 34th Hawaii International Conference on System Sciences*. IEEE. Retrieved December 10, 2003, from <http://csdl.computer.org/comp/proceedings/hicss/2001/0981/03/09813046.pdf>

Object Management Group. (2003). Unified Modeling Language Specification (Version 1.5) [Language specification]. Retrieved from <http://www.omg.org/technology/documents/formal/uml.htm>

Oracle Corporation. (2003a). Oracle database application developer's guide: Object-relational features. 10g release 1 [Computer software manual]. Retrieved from [http://otn.oracle.com/pls/db10g/portal.portal\\_demo3?selected=3](http://otn.oracle.com/pls/db10g/portal.portal_demo3?selected=3)

Oracle Corporation. (2003b). Oracle database. Concepts. 10g release 1 [Computer software manual]. Retrieved from

[http://otn.oracle.com/pls/db10g/portal.portal\\_demo3?selected=3](http://otn.oracle.com/pls/db10g/portal.portal_demo3?selected=3)

Rational Software Corp. (2000). *Using Rose Oracle8*. Santa Clara, CA: Rational Software Corp.

Rational Software Corp. (2001). *Using Rose data modeler*. Santa Clara, CA: Rational Software Corp. Retrieved December 10, 2003, from <ftp://ftp.software.ibm.com/software/rational/docs/documentation/manuals/rose.jsp>

Stonebraker, M., & Brown, P. (1999). *Object-relational DBMSs: Tracking the next great wave*. Morgan Kaufman.

Zendulka, J. (2001). Object-relational modeling in UML. *Proceedings of the Conference Information Systems Modelling* (pp. 17-24).

## KEY TERMS

**Collection Type:** A composite value comprising elements of some data types. SQL:1999 supports arrays, which are ordered and unbounded sets of elements. Another collection type, not supported by the standard, is a nested table.

**Object-Relational Data Model:** Extends the relational data model by providing a richer type system, including complex data types and object orientation.

**Object-Relational Modeling:** Modeling of an object-relational database schema. It requires using model elements that are available in neither classic data models nor object models.

**REF:** A data type value which references a row in a referenceable table (or object, in Oracle).

**Referenceable Table:** A table whose row can be referenced by REF type values. The table is based on a structured user-defined type and comprises one extra column containing row (or object) identifiers generated automatically when a new row is inserted into the table. In Oracle, such a table is called an *object table*.

**SQL:1999:** The most recent major release of the SQL standard, published in 1999, which is based on an object-relational model. Thus, it is a standard database language for object-relational databases.

**UML:** Unified Modeling Language (UML) is the industry standard modeling language (mainly graphical) for visualizing, specifying, constructing, and documenting the artifacts of software systems.

**UML Profile:** A predefined set of stereotypes, tagged values, constraints, and notation icons that collectively

specialize and tailor the UML for a specific domain or process.

**UML Stereotype:** One of UML extensibility mechanisms. It is an extension of the vocabulary of the UML that allows creating new kinds of building blocks that are derived from existing ones.

**User-Defined Type:** A named data type defined by a user. It can contain a list of attributes, in which case it is said to be a structured type (or object type, in Oracle). It is an abstraction of a real-world entity. It can also provide explicitly defined methods that implement operations with the entity.

# Online Data Mining

**Héctor Oscar Nigro**

*Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina*

**Sandra Elizabeth González Císaro**

*Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina*

## INTRODUCTION

Several approaches for intelligent data analysis are not only available but also tried and tested. Online analytical processing (OLAP) and data mining represent two of the most important approaches. They mainly emphasize different aspects of the data and allow deriving of different kinds of information. So far, these approaches have mainly been used in isolation (Schwarz, 2002).

OLAP is a powerful data analysis method for multidimensional analysis of data warehouses. Data mining is the extraction of interesting (i.e., nontrivial, implicit, previously unknown, and potentially useful) information or patterns from data in large databases (Fayad, Piatetsky-Shapiro, Smyth, & Uthurusamy, 1996; Han & Kamber, 2001).

OLAP is more frequently used for verification of the existing hypothesis. Data mining tries to generate such a hypothesis by uncovering hidden patterns. It is essentially an inductive process. It means that OLAP and data mining should be used together to complement each other (Schwarz, 2002).

Motivated by the popularity of OLAP technology, Han developed an online analytical mining (OLAM) mechanism for multidimensional data mining.

This paper is organized in the following way: First, the background section, in which OLAP and data mining are introduced. Second, the main section is divided into the following subsections: Definition, Online Mining—Expected Characteristics, Olam Architecture, Olam and Complex Data Types, and OLAM System—Essential Functionality. Then, Future Trends in the area. Finally the Conclusions, References, Key Terms.

## BACKGROUND

Basically, an OLAP system must show multidimensional data with dimensions, measures, and hierarchies through the cube's paradigm. The multidimensional model simplifies for users the process of making complex queries, modifying information in a report, swapping between aggregated and detailed data, selecting part of the data, and so forth (Han & Kamber, 2001; Lenz & Shoshani, 1997).

Selected functionalities include the following:

- drilling down
- rolling up
- dice & slice
- pivoting
- cubing

There are three types of OLAP systems:

- **Rolap:** Relational OLAP works on normalized data; the classic models are star and snowflake. Its characteristic is having few storing spaces but with the disadvantage of presenting the information more slowly.
- **Molap:** Multidimensional OLAP works with precalculated data for the optimization of response time. It is very fast but it uses more space than Rolap.
- **Hybrid:** Hybrid OLAP combines the benefits of both.

Data mining is an especially interesting and very complex task, as defined in the previous section, and it is also a confluence of multiple disciplines (see Table 1).

A good data mining query language will support ad hoc and interactive data mining. Ad hoc query-based data mining is better when users want to examine various data portions with different constraints. Database queries can be answered intelligently using concept hierarchies, data mining results, or data mining techniques.

The major functions of data mining follow (Han, Chee Sonny, & Chiang Jenny, 1999; Han & Kamber, 2001):

1. **Characterization:** Generalizes a set of task-relevant data into a generalized data cube, which can then be used to extract different kinds of rules or be viewed at multiple levels of abstraction from different angles. In particular, it derives a set of characteristic rules, which summarize the general characteristics of a set of user's specified data (called the target class).
2. **Comparison:** Mines a set of discriminant rules that summarize the features, which distinguish the



Table 1. A list of topics related to data mining

- Database systems, data warehouse, and OLAP
- Statistics
- Machine learning
- Visualization
- Information science
- High performance computing
- Business and application domain knowledge expertise
- Other disciplines: neuronal networks, mathematical modeling, information retrieval, pattern recognition, and others (Han, 2000)

class being examined (the target class) from other classes (called contrasting classes).

3. **Classification:** Analyzes a set of training data (i.e., a set of objects whose class label is known) and constructs a model for each class based on the features in the data. A set of classification rules is generated by such a classification process, which can be used to classify future data and develop a better understanding of each class in the database.
4. **Association:** Discovers a set of association rules at multiple levels of abstraction from the relevant set(s) of data in a database.
5. **Prediction:** Predicts the possible values of some missing data or the value distribution of certain attributes in a set of objects. This involves finding the set of attributes relevant to the attribute of interest (by some statistical analysis) and predicting the value distribution based on the set of data similar to the selected object(s).
6. **Cluster analysis:** Groups a selected set of data in the database or data warehouse into a set of clusters to ensure the interclass similarity is low and the intra-class similarity is high.
7. **Time series analysis:** Performs data analyses for time-related data in databases or data warehouses, including similarity analysis, periodicity analysis, sequential pattern analysis, and trend and deviation analysis.

## ONLINE DATA MINING: MAIN FEATURES

### Definition

*Online analytical mining (OLAM; also called OLAP mining) is among the many different paradigms and architectures for data mining systems. It integrates OLAP with data mining and mining knowledge in multidimensional databases. (Han, 1997)*

## Online Mining: Expected Characteristics

The following features are important for successful OLAM (Han, 1998a):

- Ability to mine anywhere
  - Carve any portion of a database by drilling, dicing, filtering, and so forth, for mining
  - Drill through to raw data and mining
- Support of multifeature cubes and cubes with complex dimensions and measures
  - Discovery-driven exploration of multifeature cubes
  - Spatial and multimedia dimensions/measures
- Cube-based mining method
  - Multiminig tasks and multilevel mining
- Selection and addition of data mining algorithms
  - Different algorithms may generate different results
  - Standard APIs allow users to develop their own algorithms
- Integration of data mining functions
  - First dicing, then classification, then association, and so forth
- Ad-hoc query-based (constraint-based) mining
- Fast response and high performance online
- Visualization tools
  - Data and knowledge visualization tools
- Extensibility

## Online Data Mining

- Integration with statistical package, extend to spatial, multimedia, Web

It is a promising direction due to the following (Han, 1998b):

- High quality of data in data warehouses
- Available information processing infrastructure surrounding data warehouses
- OLAP-based exploratory data analysis
- Online selection of data mining functions

## OLAM Architecture

In a similar way to how OLAP engine performs online analytical processing, an OLAM engine performs analytical mining in data cubes. Work with the data cube in the data analysis is performed through an API cube, and a metadata directory guides the data cube access. Furthermore, a data mining process might disclose that the dimensions or measures of a constructed cube are not appropriated for data analysis. Here, a refined data cube design could improve the quality of data warehouse construction. Moreover, in some data mining processes, at least some of the data may require exploration in great detail. The interaction of multiple data mining modules with an OLAP engine can assure mining is easily performed anywhere in a data warehouse (Han, 1997; Han et al., 1998).

Traditional data cube queries compute simple aggregations at multiple granularities. Moreover, traditional data cubes support only dimensions of categorical data and measures of numerical data. Support of such nontraditional data cubes will enhance the power of data mining. Cube-based data mining methods should be the foundation of the OLAM mechanism. One particular OLAM strength is the interaction of multiple data mining and OLAP functions; another one is selecting a set of data mining functions. For example, an OLAM system might be integrated with a statistical data analysis package, or it might be extended for spatial data mining, text mining, financial data analysis, multimedia data mining, or Web mining (Han et al., 1996; Han, 1998a, 1998b, 1998c; Han et al., 1998). Special attention should be paid to efficient

implementation of the OLAM mechanism (see Table 2; Han et al., 1999).

An OLAM system might well integrate a variety of data mining modules through different kinds of data cubes and visualization tools. High-performance data cube technology is critical to OLAM in data warehouses. Moreover, effective data mining requires the support of nontraditional data cubes with complex dimensions and measures, in addition to the on-the-fly computation of query-based data cubes and the efficient computation of multifeatured data cubes. These have the need of further development of data cube technology (Han, 1998b, 1998c).

While most data mining needs are query or constraint based, OLAM requires fast response to data mining requests. The organization should also explore such constraint-based mining in other data mining tasks. There is a wide range of data mining algorithms. The OLAM paradigm offers the user freedom to explore and discover knowledge by applying any sequence of data mining algorithms with data cube navigation (Han et al., 1999).

## OLAM and Complex Data Types

A generalization based data mining method can generalize complex objects, construct a multidimensional object cube, and perform analytical mining in such an object cube. Spatial data mining can be performed in a spatial database as well as in a spatial data cube (Han, 1998c).

By the method of OLAM of text and multimedia data, text/multimedia data cubes are built, whereupon the cube-based relational and spatial mining techniques are extended towards mining text and multimedia data. The process includes construction of a Web Log, and the performance of time-related, multi-dimensional data analysis and data mining (Han, 1999, 2000).

The work of Pei (2003) exhibits a general model for OLAP with complex types data, called Golap. Some ideas can be appropriate to online data mining architecture.

## OLAM System: Essential Functionality

By integration of OLAP and data mining, OLAP mining facilitates flexible mining of interesting knowledge in

Table 2. Review of OLAM principle implementation issues

- Modularized design and standard APIs
- Support of online analytical mining by high-performance data cube technology
- Constraint-based online analytical mining
- Progressive refinement of data mining quality
- Layer-shared mining with data cubes
- Bookmaking and backtracking techniques

data cubes because data mining can be performed at multidimensional and multilevel abstraction space in a data cube. Cubing and mining functions can be interleaved and integrated to make data mining a highly interactive and interesting process. Here, we first examine what the desired OLAP mining functions (Han, 1997, 1998c) are:

1. **Cubing then mining:** With the availability of data cubes and cubing operations, mining can be performed on any layers and any portions of a data cube. This means that one can first perform cubing operations to select the portion of data and then set the granularity level before a data mining process starts.
2. **Mining then cubing:** This means that data mining can be first performed on a data cube and then particular mining results can be analyzed further by cubing operations. Then, for each obtained class, such as the high profit class, cubing operations can be performed (e.g., drill-down to detailed levels and examine its characteristics).
3. **Cubing while mining:** A flexible way to integrate mining and cubing operation is to perform similar mining operations at multiple granularities by initiating cubing operations during mining. By doing so, the same data mining operations can be performed on different portions of a cube or at different abstraction levels.
4. **Backtracking:** To facilitate interactive mining, a mining process should allow to backtrack one or a few steps or backtrack up to a preset mark and then explore alternative mining paths.
5. **Comparative mining:** A flexible data miner should allow comparative data mining, that is, the comparison of alternative data mining processes.

It is possible to have other combinations in OLAP mining; for example, it could be performed as “mining then mining,” such as to first perform classification on a set of data and then to find association patterns for each class (Han, 1997).

## FUTURE TRENDS

The future for research and development is based on Aggarwal (2001); Aref Walid, Elfeky Mohamed, and Elmagarmid Ahmed (2004); Boulicaut, Marcel, and Rigotti (2001); Dong et al. (2003); Han (1998b, 1999); Radivojevic, Cvetanovic, Milutinovic, and Sievert (2003); Schwarz (2002):

Aggarwal (2001) shows the benefits of the interactive visual approaches. These can be applied to online data

Table 3. A summary of questions for research and development

- Complete integration with data warehouse, OLAP, and relational technology
- Scalability: efficient algorithms, parallel/distributed and incremental mining
- Ad-hoc mining query language and its optimization
- Multiple, integrated data mining functions and methods
- Mining on new kinds of data: time-series data, text, multimedia, spatial, and Web
- Visual data mining and knowledge visualization
- Application exploration
- Interactive, exploratory data mining environment

mining, to enable the user evaluate and understand the quality of his or her discoveries.

## CONCLUSION

Most data mining tools must work on integrated, consistent, and clean data. This requires costly preprocessing for data cleaning, data transformation, and data integration. Therefore, a data warehouse constructed by valuable source of high-quality data for OLAP and data mining. Data mining may also serve as a valuable tool for data cleaning and data integration.

Effective data mining requires *exploratory data analysis*. OLAM provides facilities for data mining on different subsets of data and at different levels of abstraction. It does this by drilling, pivoting, filtering, dicing, and slicing on a data cube and on intermediate data mining results. This, together with data-knowledge visualization tools, can greatly enhance the power and flexibility of exploratory data mining.

By integrating OLAP with multiple data mining functions, OLAM provides users with the flexibility to select desired data mining functions and dynamically swaps data mining tasks.

From our point of view, one important application area of online data mining is constituted by the scientific databases. Specially, those referenced to socioeconomical aspects, such as census, demographic databases and life forms.

## REFERENCES

Aggarwal, C. (2001). Interaction towards effective and interpretable data mining by Visual. *ACM SIGMOD Exploration*, 3(2), 11-22.

Aref Walid, G., Elfeky Mohamed, G., & Elmagarmid Ahmed, K. (2004). Incremental, online, and merge mining of partial periodic patterns in time-series databases. *IEEE Transactions on Knowledge and Data Engineering*, 16(3), 332-342.

- Boulcaut, J., Marcel, P., & Rigotti, C. (2001). Query driven knowledge discovery via OLAP manipulations. Retrieved March 2004 from <http://lisi.insa-lyon.fr/~jfboullic/bda01.ps>
- Dong, G., Han, J., Lakshmanan, L., Pei, J., Wang, H., & Yu, P. (2003). Online mining of changes from data streams: Research problems and preliminary results. *Proceedings of the 2003 ACM SIGMOD Workshop on Management and Processing of Data Streams. In cooperation with the 2003 ACM-SIGMOD International Conference on Management of Data (SIGMOD'03)*, San Diego, California, June 8. Retrieved March 2004 from [http://www.cse.buffalo.edu/faculty/jianpei/publications/minechange\\_mdps03.pdf](http://www.cse.buffalo.edu/faculty/jianpei/publications/minechange_mdps03.pdf)
- Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., & Uthurusamy, R. (1996). *Advances in knowledge discovery and data mining*. Menlo Park, CA: AAAI Press.
- Han, J. (1997, October). OLAP mining: An integration of OLAP with data mining. *Proceedings of the 1997 IFIP Conference on Data Semantics (DS-7)*, Leysin, Switzerland. Retrieved March 2004 from <ftp://ftp.fas.sfu.ca/pub/cs/han/kdd/olapm.ps>
- Han, J. (1998a, August). *Data mining and data warehousing research in Simon Fraser University*. Presentation at the IFIP W2.6 Workshop, New York. Retrieved from <ftp://ftp.fas.sfu.ca/pub/cs/han/slides/ifip98.ppt>
- Han, J. (1998b, December). *Towards on-line analytical mining: An overview of data warehousing and data mining*. An Invited Talk in BC Hydro Conference, Vancouver, British Columbia, Canada. Retrieved November 2003 from <ftp://ftp.fas.sfu.ca/pub/cs/han/slides/bchydro98.ppt>
- Han, J. (1998c, December). *Towards on-line analytical processing and data mining for electronic commerce*. Workshop on Technological Challenges on Electronic Commerce, CASCON'98 (CASCON-98), Toronto, Canada. Retrieved March 2003 from <ftp://ftp.fas.sfu.ca/pub/cs/han/slides/ibmec98.ppt>
- Han, J. (1999, November). *Why is data mining the next frontier of high performance computing?* Panel discussion at Super Computing '99, Portland, Oregon. Retrieved November 2003 from <ftp://ftp.fas.sfu.ca/pub/cs/han/slides/sc99.ppt>
- Han, J. (2000, January). *From DBMiner to WebMiner: What is the future of data mining?* Talk at Commerce, UBC, BC. Retrieved June 2003 from <ftp://ftp.fas.sfu.ca/pub/cs/han/slides/ubc00.ppt>
- Han, J., Chee Sonny, H. S., & Chiang Jenny, Y. (1998). Towards on-line analytical mining in large databases. Retrieved December 2000 from <http://citeseer.ist.psu.edu/cache/papers/cs/27046/http://zszszszwww-faculty.cs.uiuc.edu/zszszszpdfzszsum98.pdf/towards-on-line-analytical.pdf/>
- Han, J., Chee Sonny, H. S., & Chiang Jenny, Y. (1999). Issues for on-line analytical mining of data warehouses. Retrieved March 2004 from <http://www.acm.org/sigs/sigmod/disc/disc99/disc/dmkd/han.pdf>
- Han, J., Fu, Y., Wang, W., Chiang, J., Gong, W., Koperski, K., et al. (1996, August). DBMiner: A system for mining knowledge in large relational databases. *Proceedings of the 1996 International Conference on Data Mining and Knowledge Discovery (KDD '96)*, Portland, Oregon. Retrieved March 2004 from [ftp://ftp.fas.sfu.ca/pub/cs/han/slides/kdd96\\_slides.ppt](ftp://ftp.fas.sfu.ca/pub/cs/han/slides/kdd96_slides.ppt)
- Han, J., & Kamber, M. (2001). *Data mining: Concepts and techniques*. San Francisco: Morgan Kaufmann.
- Lenz, H., & Shoshani, A. (1997). Summarizability in OLAP and statistical data bases. Retrieved August 2002 from <http://www.lbl.gov/~arie/papers/summarizability.SSDBM97.ps>
- Pei, J. (2003). A general model for online analytical processing of complex data. *Proceedings of the 22nd International Conference on Conceptual Modeling (ER '03)*, Chicago, IL, October 13-16. Retrieved April 2004 from [http://www.cse.buffalo.edu/faculty/jianpei/publications/golap\\_er03.pdf](http://www.cse.buffalo.edu/faculty/jianpei/publications/golap_er03.pdf)
- Pei, J., & Han, J. (2002). Constrained frequent pattern mining. *ACM SIGMOD Exploration*, 4(1), 32-39.
- Radivojevic, Z., Cvetanovic, M., Milutinovic, V., & Sievert, J. (2003). Data mining: A brief overview and recent IPSI research. *Annals of Mathematics, Computing & Teleinformatics*, 1(1), 84-90.
- Schwarz, H. (2002). *Integration von data mining und online analytical processing: Eine analyse von datenschemata, systemarchitekturen und optimierungsstrategien*. Doctoral dissertation, Institut für Parallele und Verteilte Systeme der Universität Stuttgart, Stuttgart, Germany. Retrieved December 2003 from <http://elib.uni-stuttgart.de/opus/volltexte/2003/1447/pdf/schwarz.pdf>

## KEY TERMS

**Bookmaking and Backtracking Techniques:** The OLAM paradigm offers the user complete freedom to explore and discover knowledge by applying any se-

quence of data mining algorithms with data cube navigation. Often, a user has a choice of many alternatives when traversing from one data mining state to another. It would be useful if she or he can set bookmarks: If a discover path proves uninteresting, she or he can return to a previous state and explore other alternatives. Efficient support of such marking and backtracking mechanisms will protect users from being “lost in the OLAM space” (Han, 1997; Han et al., 1999, p. 4).

**Constraint-Based Online Analytical Mining:** Online analytical mining requires fast response to data mining requests whereas most data mining requests are query based, or constraint based. This requires mining not only be performed with a limited scope of data, confined by queries and constraints, but also adopt efficient, constraint-based data mining algorithms. For example, many constraints involving set containments or aggregation functions can be pushed deeply into the association rule mining process. Such constraint-based mining should be explored in many other data mining tasks (Han et al., 1999). A good proposal of constrained frequent pattern mining can be found in Pei and Han (2002).

**Cube:** A data structure of aggregated values summarized for a combination of preselected categorical variables (e.g., number of items sold and their total cost for each time period, region, and product). This structure is required for high-speed analysis of the summaries that is done in online analytical processing (OLAP). Also called a multidimensional database, or MDDB.

**Drill Down:** The process of navigating from a top-level view of overall sales down through the sales territories, to the individual salesperson level. This is a more intuitive way to obtain information at the detail level. Drill-

down levels depend on the granularity of the data in the data warehouse. Roll up is the opposite function.

**Filters:** Saved sets of chosen criteria that specify a subset of information in a data warehouse.

**Granularity:** The level of detail of the facts stored in a data warehouse, or a concept in the database.

**Layer-Shared Mining with Data Cubes:** Because each dimension in a data cube represents an organized layer of concepts, data mining can be performed by first examining the high levels of abstraction and then progressively deepening the mining process towards lower levels of abstraction. This will save the efforts of indiscriminate examination of all the concepts at low level. It is important to explore this optimization at mining other kinds of knowledge (Han et al., 1999).

**Support of Online Analytical Mining by High-Performance Data Cube Technology:** High-performance data cube technology is critical to online analytical mining in data warehouses. There have been many efficient data cube computation techniques developed in recent years that helps efficient construction of large data cubes. However, since a mining system may need to compute the relationships among a great number of dimensions or examine the fine details, but such data may not always be materialized beforehand, it will be necessary to dynamically compute portions of data cubes on the fly. Moreover, besides on-the-fly computation of query-based data cubes, the efficient computation of multifaceted data cubes, and the support of nontraditional data cubes with complex dimensions and measures, they are crucial to effective data mining. Therefore, further development of data cube technology will provide enhanced support to OLAM (Han, 1997; Han et al., 1999).



# Ontological Assumptions in Information Modeling

**John M. Artz**

*The George Washington University, USA*

## INTRODUCTION

Information modeling is a technique by which a database designer develops a conceptual model of a database depicting the entity classes that will be represented in the database. There are three competing ontological assumptions that guide the modeling process. The broadest characterization of these assumptions is realism vs. conceptualism, with social realism occupying a middle ground. The realist believes that object classes exist in the real world, waiting to be discovered. The conceptualist believes that object classes are constructed in the mind of the modeler, based on observations about the application domain and the objectives of the information model. The social realist believes that classes exist as shared meanings among stakeholders in an application domain. This article explores these assumptions and then reviews selected literature in information modeling to determine which assumptions are held by key authors. It concludes that most authors hold inconsistent views, and this inconsistency provides some important insights into information modeling while presenting serious problems for practitioners and students of information modeling.

## BACKGROUND

Perhaps one of the most perplexing problems in information modeling is the ontological status of entity classes. This raises the question, Do entity classes exist in the world, or are they constructed in the mind of the modeler? This seemingly esoteric question is important because the way in which one answers it has a significant impact on how one approaches the process of information modeling.

If entity classes exist in the world, independent of the mind of the observer, then the job of the information modeler is to discover those classes and record them in an information model. Hence, information modeling is a discovery process rather than a constructive process. If two modelers examine a domain and come up with different models, then one is wrong. One or the other (or possibly both) must bring their models into conformance with the real world. An information model can be

validated by ensuring its conformance with the real world. This is the realist position. To the realist, the challenges in information modeling are how to discover the existing entity classes and accurately represent those classes in an information model. Validation is not a problem for the realist because a model is valid if it correctly represents the real world. Realism can be detected when writers use the phrases “real world” or “as they exist in the real world” or, more subtly, when they refer to “natural classes” or “natural data relationships.”

If classes do not exist in the world, then it is the job of the information modeler to construct them. Thus, information modeling becomes a process of construction rather than discovery. The conceptualist position holds that classes exist only in the mind of the observer and are constructed according to objectives (probably not explicit) that guide the process of abstraction. The modeler selects certain facts from the application domain and constructs classes based on individual objects with similar attributes. Conceptualists believe that different models of an application domain cannot be determined to be correct or not correct. They can only be more or less useful for meeting the model objectives. This leads to problems in both construction and validation of the information model. Construction is difficult because most of the literature on information modeling focuses on description of entities rather than construction of them. The literature is strangely silent on how to construct a set of entity classes to meet a set of modeling objectives. Validation is also a problem because the model cannot be compared with entities existing in the real world, it can be evaluated only with respect to the objectives of the model, again an area in which the literature is strangely silent. The conceptualist position creates serious problems for information modelers because it requires that modeling objectives be defined before model construction, and it requires some method of evaluating a model with respect to a set of objectives. Conceptualism can be detected in the literature when authors talk about “abstraction,” “the problem to be solved,” “objectives,” or the possibility of “multiple representations” or “multiple models.”

An intermediate position is social realism, which assumes that entity classes exist as shared meanings within a social context. This is a realist position in that the classes exist independent of the mind of the modeler.

Presumably if several modelers were to examine the same application domain they would eventually discover the same shared meanings and hence would produce the same information model. Validation is less problematic under the social realist position in that the model can be compared with the social reality, that is, the model agrees with what people in the application domain believe to be the entity classes or it does not. The social realist discovers the entity classes by talking with users and recording their usage of key words. Consensus is an important factor in the social realist approach because social reality is a shared understanding. If people do not agree on meanings, then the realism assumption breaks down because different modelers may very well come away with different understandings, depending on who they spoke to. Domain experts are important to the social realist position because the domain expert is the gatekeeper to the social reality. Social realism can be detected in the literature when authors talk about “language,” “shared meanings,” “domain or subject matter experts,” or modeling as a process of “consensus.”

Realism is a shaky assumption from a philosophical perspective, but desirable from a pragmatic perspective. If entity classes exist in the world, where do they reside? Although there are ample *instances* of an entity class, nobody has ever seen the class itself, nor will they. Entity classes exist only in the mind of the observer and have no real existence in the application domain. The modeler examines the application domain and, through a cognitive process of abstraction, derives a set of entity classes. Yet this process of abstraction is poorly understood and difficult to explain, so modelers act as though the classes actually exist in the world and are being discovered. From a pragmatic perspective, realism is a desirable assumption because it reduces the class construction process to one of simple discovery and provides an easy means of validation by requiring that the model simply conform to the real world.

Conceptualism is a much more justifiable position from a philosophical perspective, yet a nightmare in practice. Conceptualism recognizes the role of modeler and his or her cognition in the class construction process. Yet in practice it presents some severe problems. Since classes are constructed, how does the process of construction work? What criteria are used in class construction? Once classes are constructed, how do we know that the right classes have been constructed? One answer is that classes are constructed by grouping objects with similar attributes. But that position raises the question of whether or not attributes exist in the world and opens up, once again, the three positions just described with regard to attributes. Another answer might be to say that the classes are right if they meet the objectives of the model, but that answers the question by

raising two more: How do we define modeling objectives and how do we determine if a set of classes meets those modeling objectives?

Most authors nod towards conceptualism, using terms such as *problem solving* or *multiple models* but back off when it comes to the actual process of modeling, where they will often fall back to a realist position by talking about modeling “the real world.” Recognizing the faultiness of the realist position, several authors have adopted an intermediate position of social realism. The more rigorous ones adopt social realism with respect to attributes, but the best that can be said is the literature is confusing and few authors have taken and articulated a consistent philosophical position.

## ANALYSIS OF THE LITERATURE

The ontological assumptions made by practitioners are rarely articulated. They are more often manifest in their behavior. Practitioners may even claim to hold one belief while acting as though they held a conflicting view. Hence, in order to gain a sense of the variety of assumptions that are held in the field of information modeling, it is necessary to look at the recorded literature—widely read texts and papers—to see what assumptions are being put forth.

Peter Chen’s (1976) original article on the entity-relationship model begins by establishing a clearly realist perspective: “The entity-relationship model adopts the more natural view that the real world consists of entities and relationships. It incorporates some of the important semantic information about the real world” (p. 9-10). This perspective is picked up by later authors. Andleigh and Gretzinger (1992) claimed “the Entity Model describes the real-world relations for the information system” (p. 383), while Teorey (1990) referred to the model as including “the natural data relationships,” again a strong indication of realism. Yet only a couple of paragraphs later in Chen’s paper, he refers to a conceptual data model as “information concerning entities and relationships which exist in our minds” (p. 10) and “conceptual objects in our minds” (p. 14), showing a clearly conceptualist perspective. In discussing whether a given object should be an entity or a relationship, Chen defers to the enterprise administrator, who should decide so that “the distinction is suitable for his environment” (p. 10), suggesting an objectives-driven conceptualist view or a social-realist view, depending on the meaning of the word *suitable*. He goes on a bit later to say, “If we know an entity is in the entity EMPLOYEE, then we know that it has the properties common to other entities in the entity set” (p. 11), suggesting extreme class realism. Since nothing is said, in the

paper, regarding how to construct entities, the reader is forced to defer back to the realist position and use the real world to guide the discovery and validation processes.

Only two years after Chen's (1976) original article, Kent (1978) thoroughly devastated the realism assumption in *Data and Reality* by raising question after question that could not be answered from the realist's perspective. "There is no natural set of categories" he said. "The set of categories to be maintained in an information system must be specified for that system" (p. 13). He goes on to say, "If we really did want to define what a data base modeled, we'd have to start thinking in terms of mental reality rather than physical reality. Most things are in the data base because they 'exist' in people's minds, without having any 'objective' existence." This is about as clear of a statement of conceptualism as one can find. It is unfortunate that this work raised many, many more questions than it answered. This, coupled with the fact that it is difficult to follow, reduced its impact on the practice of modeling. Later, Kent (1986) adopted a more conservative, fact-based approach that trades entity realism for attribute realism and allows class construction based on entities with similar attributes.

Nijssen and Halpin (1989) also adopted a fact-based approach, but give the reader mixed signals with regard to their ontological assumptions. They recognize that classes are constructed, but see this construction as guided by facts gathered from the Universe of Discourse. The ontological status of the Universe of Discourse is unclear. They say, "Recall that the UoD is the portion of the (typically) real world relevant to our application" (p. 35), which suggests realism. Yet, they go on to say, "Recall that entities are the basic objects or things that we want to talk about" (p. 37), which suggests that the Universe of Discourse is a social construct based on language usage. This view is reinforced by the statement that "The UoD expert or domain expert is familiar with the application area, and can clarify any doubtful aspects of the UoD" (Nijssen & Halpin, 1989; p. 14). These claims of social realism are further supported by the validation process: "Our conceptual schema design procedure facilitates early detection of errors by various checking arrangements including ongoing feedback to the user by way of examples" (p. 199). Here the model is being validated against user opinions, which suggests social realism.

Conflicting ontological assumptions are not uncommon. Shlaer and Mellor (1988) reflect the same confusion. At one point they assert, "What is needed is a way to capture information so that it can be checked against the reality, rather than the different, and possibly inconsistent, 'user views' of reality" (p. 3), which reflects a realist view of information modeling and a criticism of social realism. Then, a few paragraphs later, they say, "What we need is a method by which we can lay out candidate definitions of

the conceptual entities and examine the implications of those definitions" (p. 4), which reflects a conceptualist view.

Flavin (1981) clearly stated that "the decomposition of the system of interest into its component objects is a function of the system, the observer, and their mutual interaction" (p. 38), which again reflects a conceptualist perspective. Though Flavin does provide implicit modeling objectives in terms of abstraction and classification, he does not discuss how to construct classes to meet a given set of information objectives. Instead, he describes functional, transaction, and scenario analysis as a means of discovering entities. Hence, the ontology is unclear.

But it was not until Klein and Hirschheim (1987) that the problem was grounded philosophically. Klein and Hirschheim examined both ontological and epistemological assumptions in information modeling. They identified two ontological poles that they referred to realism and nominalism. Nominalism and conceptualism are slightly different antirealist views. Nominalism asserts that things are what they are because we have named them that way. The collection of things that share a name need not have anything in common other than the fact that they share the same name. Conceptualism asserts that there are abstract mental entities, called concepts, which are abstracted from the particulars that we experience. The name that we apply to a thing is really the name of a concept of the thing.

The social realist view is problematic because, whether the social reality is stable or in a process of change, the entity classes that exist in usage may or may not be appropriately constructed to meet the objectives of the information model. Yet, as Hirschheim, Klein and Lyytinen say, "The potential of objects to model imaginary, 'ideal' or socially constructed worlds has not, unfortunately, been widely recognized in the data modeling literature" (1995, p. 62). Indeed, since conceptualism requires the construction of imaginary, possible worlds, it is scrupulously avoided by researchers in information modeling. Yet the question must be asked: Is it better to use faulty philosophical assumptions that are easier to work with, or sound philosophical assumptions that may create problems in practice?

Rumbaugh et al. (1991) come the closest of any of the nonphilosophers to correctly articulate a consistent set of assumptions. They claim that "in the real world an object simply exists," which is a safe assumption for modeling purposes, since it refers to the independent existence of the instance rather than the class. They go on to say "Classification means that objects with the same data structure (attributes) and behavior (operations) are grouped together in a class" (p. 2). This is clearly a conceptualist perspective since the grouping is

a result of the process of classification. To emphasize the point further, they state that “a class is an abstraction that describes properties important to an application and ignores the rest. Any choice of classes is arbitrary and depends on the application.” This is a clear statement of the conceptualist position. In defense of the other authors, conceptualism is more obvious in pure object-oriented systems because the objectives of hierarchy and reusability are much clearer.

On a closing note, it is interesting that practitioners are sometimes aware of deeper philosophical issues, but their concern is often brief. Coad and Yourdon (1990), who published one of the first books on object-oriented analysis, sum up the practitioner’s scant concern:

*As authors, it would be intellectually satisfying if we could report that we studied the philosophical ideas behind methods of organization, from Aristotle and Socrates to Descartes and Kant. Then, based on the underlying methods human beings use, we could propose the basic constructs essential to a requirements analysis method, and in particular to OOA [object-oriented analysis]. But in truth, we cannot say that, nor did we do it. (p. 16)*

## FUTURE TRENDS

It is unlikely that this confusion in ontological assumptions will just go away by itself. Until information modeling is put on a firm philosophical foundation, we will continue to see confusion among authors and inconsistencies within specific treatments of the topic.

## CONCLUSION

The best that can be said regarding the ontological assumptions underlying information modeling is that there is a great deal of confusion. Few authors really hold the realist position, as can be seen in their constant references to conceptualist ideas. Yet abandoning the realists’ position creates serious methodological problems in discovery and validation. Some have adopted social realism in a retreat from realism. This position handles the discovery and validation problems but is only slightly more tenable than realism. Social realism assumes that the classes that exist in the shared meanings of the stakeholders of the application domain are exactly the classes needed to meet the objectives of the information model. With luck this may be true, but it is unlikely. First, socially constructed classes are more likely to meet the objective of intellectual economy than any processing or information objectives. Second, social realism assumes

that the application domain is static and that classes which have been useful in discourse in the past will be useful for information derivation in the future. It seems that conceptualism is the only justifiable ontological position. Unfortunately, conceptualism requires that modeling objectives be explicitly articulated and that models be constructed to meet objectives and evaluated with respect to those objectives. There is much work to be done here, but since it appears that conceptualism is the only sound foundation for information modeling, it is probably time to get started.

## REFERENCES

- Andleigh, P., & Gertzinger, M. (1992). *Distributed object-oriented data-systems design*. Englewood Cliffs, NJ: Prentice Hall.
- Artz, J. (1997). A crash course in metaphysics for the database designer. *Journal of Database Management*, 8(4).
- Chen, P. (1976). The entity relationship model: Towards a unified view of data. *Association of Computing Machinery Transactions on Database Systems*, 1(1).
- Coad, P., & Yourdon, E. (1990). *Object-oriented analysis*. Englewood Cliffs, NJ: Prentice Hall.
- Flavin, M. (1981). *Fundamental concepts of information modeling*. New York: Yourdon Press.
- Hirschheim, R., & Klein, H. K. (1989). Four paradigms of information systems development. *Communications of the ACM*, 32(10), 1199-1216.
- Hirschheim, R., Klein, H. K., & Lyytinen, K. (1995). *Information systems development and data modeling: Conceptual and philosophical foundations*. Cambridge, UK: Cambridge University Press.
- Kent, W. (1978). *Data and reality*. New York: North Holland.
- Kent, W. (1979). Limitations of record-based information models. *ACM Transactions on Database Systems*, 4(1) 107-131.
- Kent, W. (1986). The realities of data: Basic properties of data reconsidered. In T.B. Steele, Jr., & R. Meersman (Eds.), *Database semantics (DS-1)*. Elsevier Science.
- Klein, H. K., & Hirschheim, R. A. (1987). A comparative framework of data modelling paradigms and approaches. *The Computer Journal*, 30(1), 8-15.
- Nijssen, G. M., & Halpin, T. A. (1989). *Conceptual schema and relational database design: A fact oriented approach*. New York: Prentice Hall.



## ***Ontological Assumptions in Information Modeling***

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., & Lorenzen, W. (1991). *Object-oriented modeling and design*. Englewood Cliffs, NJ: Prentice Hall.

Shlaer, S., & Mellor, S. J. (1988) *Object-oriented systems analysis: Modeling the world in data*. Englewood Cliffs, NJ: Yourdon Press.

Teorey, T. J. (1990). *Database modeling and design: The entity-relationship approach*. San Mateo, CA: Morgan Kaufman.

Veryard, R. (1984). *Pragmatic data analysis*. Oxford: Blackwell Scientific Publications.

Veryard, R. (1992) *Information modeling*. New York: Prentice-Hall.

Woozley, A.D. (1967) Universals. In P. Edwards (Ed.), *Encyclopedia of Philosophy* (Vol. 8, pp. 194-206).

## **KEY TERMS**

**Attribute Realism:** An ontological position that the properties of entities exist in the world independent of their being perceived by the modeler.

**Conceptualism:** An ontological position that entity classes exist only in the mind of the modeler.

**Entity Realism:** An ontological position that entity classes exist in the world independent of their being perceived by the modeler.

**Nominalism:** An ontological position that things are what they are because we have named them that way, and classes are formed of objects with similar names.

**Ontology:** A branch of philosophy that attempts to determine the structure of reality. It can also refer to a classification system that organizes particular things or phenomena.

**Social Realism:** An ontological position that entity classes exist in the shared meanings of a social group.

**Universe of Discourse:** The application domain of an information model viewed from a semantic perspective.



# Ontologies and Their Practical Implementation

Gian Piero Zarri

University of Paris IV/Sorbonne, France

## INTRODUCTION: “ONTOLOGIES” AND “TAXONOMIES”

Starting from the '90s, ontologies have emerged as an important research topic investigated by several research communities (including the database community) and used especially in defining standards for data exchange, information integration, and interoperability. The word “ontology” comes from medieval philosophy, where it was used to talk about the existence of beings in the world (Guarino & Giaretta, 1995). According to its modern computer-science technical meaning (Gruber, 1993), a consensus definition says that, “Ontologies represent a formal and explicit specification of a shared conceptualization,” (p. 199) where:

- *Conceptualization* refers to an abstract model of some phenomenon/situation in the world, where the model results by the identification of the relevant concepts that characterize this particular phenomenon/situation. To avoid any hype, “concepts” can be simply understood here as the discrete, important notions that must be necessarily utilized to describe the phenomenon/situation under consideration.
- *Explicit* means that the type of concepts used and the constraints on their use are explicitly defined.
- *Formal* refers to the fact that the ontology should be machine-usable.
- *Shared* reflects the notion that an ontology captures consensual knowledge, that is, this knowledge is not private to some individual but must be accepted by a group.

A definition like this needs, however, some further discussion. Apart from the requirement of being usable on a computer, there is nothing in the previous characterization of an ontology that, for example, could not be applied also to a “taxonomy” in the classical Linnaean meaning: It is obvious, in fact, that Linnaeus’ classifications for biology were intended to give an exhaustive definition of some phenomena/situations and that they were intended to be explicit and shared. And surely a “taxon” for a notion like “mammal” is not so different from a “concept” for the

same notion. Moreover, both the concepts and the taxa are organized into a hierarchy that takes the form of a tree—of a DAG (direct acyclic graph) if multi-inheritance (see the next section) is admitted. For a (pragmatic) distinction between “taxonomies” and “ontologies,” we must then rely on the “scope” of the definitions associated with the taxa/concepts.

Speaking for simplicity’s sake, from this time onward, of concepts, in a taxonomy (and in the most simple types of ontologies) the *implicit* definition of a concept derives simply by the fact of being inserted in a network of specific/generic relationships with the other concepts of the taxonomy/hierarchy. This means that a concept like *company\_* is defined by the fact of being, at the same time, a *specific* term of a higher order concept like *social\_body* (*company\_* is subsumed by *social\_body*) and a *generic* term with respect to a specialized concept like *computer\_company* (*company\_* subsumes *computer\_company*).

To get now a “real” ontology, we must supply also some *explicit* definitions for the concepts—or at least for a majority among them. This can be obtained, e.g., by associating a “frame” (a set of properties/attributes with associated classes of admitted values; see the next sections) with these concepts. For example, if we consider that properties useful to better specify the concept *company\_* could be, among other things, *DateOfCreation* and *DomainOfActivity*, we will associate such properties (slots) with the concept. We will also impose, at the same time, that when specific examples (“instances”) of the concept *company\_* will be created, the slot *DateOfCreation* will only be filled by instances of another concept of the hierarchy, like *date\_*, and the *DomainOfActivity* slot with instances of a second concept, like *market\_sector*. A definition mechanism like this is totally extraneous to a classical Linnaean taxonomy.

In the next sections, we will outline the main principles of the “classic” ontology’s theory as it has been developed in the artificial intelligence domain. A companion article of this encyclopedia, “Using Semantic Web Tools for Ontologies Construction” deals, on the contrary, with the new developments originated by the use of ontologies as the basic knowledge representation tool in a Semantic Web context.

## BACKGROUND

### Inheritance Hierarchies

In this subsection, we will deal with the general “architectural” issues related to the construction of well-formed hierarchies of concepts—both ontologies and taxonomies. Ontologies/taxonomies are structured as “inheritance” hierarchies, making use of the well-known IsA link—called also AKindOf (Ako), SuperC, etc.; see Figure 1. A relatively unchallenged—see, however, Brachman (1983)—semantic interpretation of IsA states that this relationship among concepts, when noted as (IsA *B A*), means that concept *B* is a *specialization* of the more general concept *A*. In other terms, *A* subsumes *B*. This assertion can be expressed in logical form as:

$$\forall x(B(x) \rightarrow A(x)) \quad (1)$$

(1) says that if any *elephant\_* (*B*) IsA *mammal\_* (*A*), and if *clyde\_* is an *elephant\_*, then *clyde\_* is also a *mammal\_*. In this section, we will adopt the convention of writing down the *concepts\_* in italics and their instances\_ (e.g., *clyde\_*, an “individual”) in roman characters. When (1) is interpreted strictly, it also implies that a given concept *B* and all its instances *must* inherit *all* the features (properties) and their values of *all* the concepts *C<sub>i</sub>* in the hierarchy that have *B* as a specialization; we speak in this case of “strict inheritance.” Note that, under the strict inheritance hypothesis, totally new properties can be added to *B* to differentiate it (specialize it) with respect to its parents.

Relation IsA is transitive: This means that, e.g., having both  $\forall x(C(x) \rightarrow B(x))$  and  $\forall x(B(x) \rightarrow A(x))$ , we can deduce from this that  $\forall x(C(x) \rightarrow A(x))$ . This property is particularly important because it allows, in an inheritance hierarchy like that of Figure 1, one to represent explicitly only the IsA relationships that associate directly two nodes (i.e., without the presence of intermediary nodes). All the residual IsA relationships are then explicitly derived only when needed: E.g., from Figure 1 and from the transitive property of IsA, we can explicitly assert that (IsA *chow\_ mammal\_*).

The necessary complement of IsA for the construction of well-formed hierarchies concerns some form of InstanceOf link, used to introduced the “instances” (concrete examples) of the general notions represented by the concepts. The difference between (IsA *B A*) and (InstanceOf *C B*) is normally explained in terms of the difference between the two options of (i) considering *B* as a subclass of *A* in the first case, operator “ $\subset$ ,” and (ii) considering *C* as a member of the class *B* in the second, operator “ $\in$ .” Unfortunately, this is not sufficient to eliminate any ambiguity about the notion of instance,

which is much more controversial than that of concept. Problems about the definition of instances concern, e.g., (i) the possibility of accepting that concepts (to the exclusion of the root) could also be considered as “instances” of their generic concepts and (ii) the possibility of admitting several levels of instances, i.e., instances of an instance. For a discussion about these problems and the possible solutions, see Bertino, Catania, and Zarri (2001, p. 138).

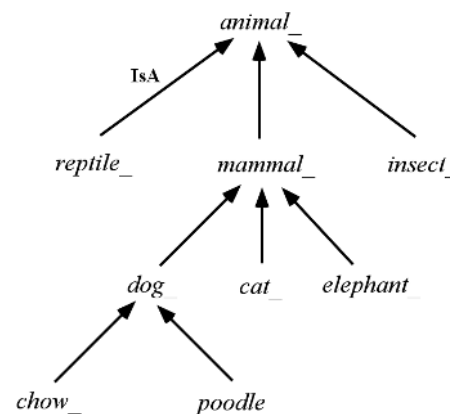
The precise definition of the meaning of InstanceOf is not the only problem that affects the construction and use of inheritance hierarchies, especially when the inheritance considered is more a “behavioral” than a “structural” one, i.e., is more interested in the actual behavior and meaning of the properties inherited than in the pure mechanical aspects of the propagation. From this point of view, we have to face two main problems: “overriding” (or “defeasible inheritance,” or “inheritance with exceptions”) and “multiple inheritance”; we will discuss overriding in some depth.

Overriding consists in the possibility of admitting exceptions to the “strict inheritance” interpretation of (1). Let us consider this group of assertions:

- a. Elephants are grey, except for royal elephants.
- b. Royal elephants are white.
- c. All the royal elephants are elephants.

Assertion (c) introduces a new concept, *royal\_elephant*, as a specialization of *elephant\_* of Figure 1. If now (InstanceOf *clyde\_ royal\_elephant*), the strict inheritance law would lead us to conclude that the property (slot; see the next subsection) ColourOf of *clyde\_* is “filled” with the value *gray\_*, but from (a) and (b) we know that the correct filler is instead *white\_*. This means that *royal\_elephant* has an “overriding property,”

Figure 1. A simple inheritance hierarchy



ColourOf or, in other terms, that the property ColourOf of *elephant\_* must not be considered as a systematically inheritable property. An at least implicit differentiation between “overriding properties” and “non-overriding properties” is then introduced in the set of properties (attributes, slots, etc.) that characterize a given concept: For *elephant\_* and all its instances and specific terms, we can say, e.g., that FormOfTheTrunk is a non-overriding property, given that its associated value will always be *cylinder\_*; ColourOf will be, on the contrary, overriding. Figure 2 visualizes the situation described in the three assertions above; the crossed line (“cancel link”) indicates that the value associated with the overriding property ColourOf has been actually changed, passing from *elephant\_* to *royal\_elephant*. Note that, in most of the systems dealing concretely with ontologies, the cancel link is not explicitly implemented, and the overriding can be systematically executed.

In a well-known article, R.J. Brachman (1985) warns about the logical inconveniences linked with the introduction of an unlimited possibility of overriding. Under the complete overriding hypothesis, the values associated with the different properties of the concepts, and the properties themselves, must now be interpreted simply as “defaults,” that is, always possible to modify. Brachman gives, in particular, a very extreme example linked with the possibility that the properties can be overridden, leaving, on the contrary, the associated values unchanged: A *giraffe\_* is then considered as an *elephant\_* where the value *cylinder\_* associated with the property TrunkOf of *elephant\_* is unchanged, but the property itself has been overridden, and it is now called NeckOf for *giraffe\_* (Brachman, 1985, pp. 85-86). It is evident that it becomes now unfeasible to make use of inheritance as a “definitional” principle, i.e., to make use of the internal structure of the different concepts—linked with the presence of

particular properties and values—to determine, e.g., whether a given concept *A* is more general or more specific than *B*. This leads, inter alia, to the impossibility of determining automatically the position of a new concept in the inheritance network and to conclude that a hypothetical *elephant\_with\_three\_legs* or a *yellow\_elephant* (this last is an unfortunate elephant suffering from hepatitis) is still an *elephant\_*. The benefits associated with the use of the inheritance hierarchies seem then close to vanishing; without endorsing completely such catastrophic conclusions, it appears clearly that an uncontrolled amount of overriding can introduce some really serious coherence problems.

Given, however, that dealing with exceptions is an evident necessity in the knowledge representation domain, artificial intelligence (AI) researchers have tried to avoid the danger of an uncontrolled use of overriding techniques by using some form of non-classical logic—in particular, Reiter’s (1980) default logic—to provide a formal semantics for inheritance hierarchies with defaults. Very briefly, a “default theory” is a pair  $(D, W)$  where  $D$  is a set of “default rules” (seen as a sort of inference rules) concerning normally the properties of the concepts, like “Typically, elephants have four legs,” and  $W$  is a set of “hard facts,” like “All elephants are mammals” (or “Margaret Mitchell wrote *Gone With the Wind*”). Formally,  $W$  is a set of first-order formulae, while a typical default rule of  $D$  can be denoted as:

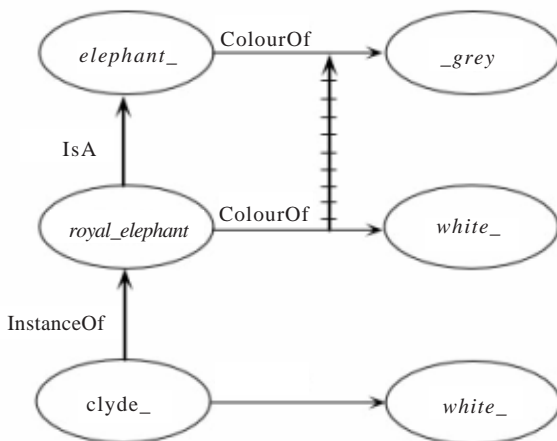
$$\frac{\alpha(x_1, \dots, x_n) : \beta(x_1, \dots, x_n)}{\gamma(x_1, \dots, x_n)} \quad ; \quad (2)$$

where  $\alpha, \beta$  and  $\gamma$  are again first-order formulae whose free variables are among  $x_1, \dots, x_n$ . The notation  $\omega(x_i)$ , used for the first-order formulae of (2), is here an abridged logical-like notation representing in general both (IsA  $x_i \omega$ ) (InstanceOf  $\omega$ ) and (HasProperty  $x_i \omega$ ). Informally, a rule like (2) means then: for any individuals  $x_1, \dots, x_n$ , if  $\alpha(x_1, \dots, x_n)$  is inferable and if  $\beta(x_1, \dots, x_n)$  can be consistently assumed, then infer  $\gamma(x_1, \dots, x_n)$ . For the previous example concerning royal elephants, (2) becomes:

$$\frac{elephant\_ (x) : gray\_ (x) \wedge \neg royal\_ elephant(x)}{gray\_ (x)} \quad . \quad (3)$$

From the informal definition above, it can be seen that, if we assume simply that (InstanceOf *clyde\_ elephant\_*), we can affirm that (ColourOf *clyde\_ gray\_*) and  $\neg$  (InstanceOf *clyde\_ royal\_elephant*) are both consistent with this assumption; hence, (ColourOf *clyde\_ gray\_*) can be inferred. In a more formal way, from the initial assumption *elephant\_*(*clyde\_*), and having veri-

Figure 2. Overriding (defeasible inheritance)

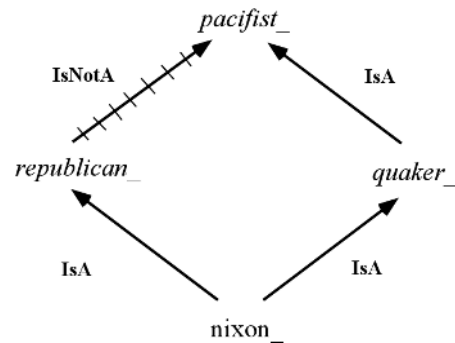


fied that  $gray\_clyde\_$  and  $\neg royal\_elephant(clyde\_)$  are consistent with the assumption, we can infer  $gray\_clyde\_$ . On the other hand, if the initial assumption is now  $royal\_elephant(clyde\_)$ , using the hard fact (IsA  $royal\_elephant\ elephant\_$ ), see Figure 1, we can be reduced again to the situation of the previous example, i.e.,  $elephant\_clyde\_$ ; in this case, however, the consistency condition  $\beta(x_1, \dots, x_n) = gray\_x \wedge \neg royal\_elephant(x)$  is violated given the initial assumption  $royal\_elephant(clyde\_)$  that “blocks” the default rule, preventing then the derivation of  $gray\_clyde\_$ .

The inheritance hierarchy of Figure 1 is a “tree,” i.e., each node (concept) has only one node immediately above it (its “parent node”), from which it can inherit the properties. In this case, the mode of transmission of the properties is called “single inheritance.” For taxonomies, this is the normal situation; in real-world ontologies, however, a concept can have multiple parents and can inherit properties along multiple paths; e.g., the  $dog\_$  of Figure 1 can also be seen as a  $pet\_$ , inheriting then all the properties of the ancestors of  $pet\_$ , pertaining maybe to a branch *private\_property* of the global inheritance hierarchy. This phenomenon is called “multiple inheritance”; the inheritance hierarchy becomes now a “tangled hierarchy” (a DAG) as opposed to a tree—a partially ordered set (poset) from a mathematical point of view. Multiple inheritance contributes strongly to the simplification of the inheritance hierarchies by eliminating the need for the duplication of the concepts and of the corresponding instances that would be required to reduce the hierarchy to a simple tree; possible examples of duplicated concepts could be *dog\_as\_valuable\_object* and *dog\_as\_carnivore\_mammal*. The use of the multiple inheritance approach can, however, give rise to conflicts when a concept inherits the same property from two (or more) different ancestors, and these properties have different values. This problem and the possible solutions are illustrated, e.g., in Bertino et al., 2001, pp. 142-144).

When defeasible inheritance (materialized by the presence of cancel links) and multiple inheritance combine, we can be confronted with very tricky situations like the notorious “Nixon diamond”; see Figure 3. In this version of the diamond, the most frequently used, we admit that it is possible to have an individual,  $nixon\_$ , as a common instance of two different concepts,  $republican\_$  and  $quaker\_$ . Several inheritance-based systems do not allow this possibility; postulating, however, the presence of an intermediate concept like *republican\_having\_quaker\_convictions*, which specializes both  $republican\_$  and  $quaker\_$  and to which we could attach the  $nixon\_$  instance, would not change the essence of the problem. If we ask now: “Is Nixon a pacifist or not?” we are in trouble, given that, as a Quaker, Nixon is (typically) a pacifist, but, as a Republican, Nixon is (typically) not a

Figure 3. Nixon diamond



pacifist. A reasoner dealing with this situation must then choose between two possible attitudes. According to a “skeptical” attitude, it will refuse to draw conclusions in ambiguous situations, and, therefore, it will emit no opinion as to whether Nixon is or is not a pacifist. According to a “credulous” attitude, the reasoner will try to deduce as much as possible, generating then all the possible “extensions” of the ambiguous situation. In the Nixon diamond case, it will then generate both the solutions,  $pacifist\_$  and  $\neg pacifist\_$ .

This problem (and similar ones) has given rise to a flood of theoretical work, which cannot be described here; see, e.g., Touretzky (1986) and Bertino et al. (2001, pp. 144-147) for a synthesis.

### Definitions of Concepts as Frames

Until now, we have considered the concepts as they were characterized solely by (i) a conceptual label (a symbolic name) and (ii) their hierarchical relationships with *all* the other concepts symbolized by the use of the IsA links. We will see now how it could be possible to associate to any concept a “structure” (a “frame”) reflecting the knowledge human beings have about (i) the intrinsic properties of these concepts and (ii) the network of relationships, other than the hierarchical one, the concepts have with each other. As already stated above, we are now totally in the “ontological” domain, and this sort of frame-based ontology can be equated to the well-known “semantic networks”; see Lehmann (1992).

Basically, a “frame” is a set of properties, with associated classes of admitted values, that is linked to the nodes representing the concepts. Introducing a frame corresponds then to establishing a new sort of relationship between the concept  $C_i$  to be defined and *some* of the other concepts of the ontology. The relationship concerns the fact that the concepts  $C_p, C_2 \dots C_n$  used in the frame defining  $C_i$  indicate now the “class of fillers” (spe-



cific concepts or instances) that can be associated with the “slots” of this frame, the slots denoting the main properties (attributes, qualities, etc.) of  $C_i$ . There is normally no fixed number of slots, nor a particular order imposed on them; slots can be accessed by their names.

To show how it can be possible to introduce a formal description of the relationships between  $C_i$  and concepts  $C_1, C_2 \dots C_n$ , let us suppose that concept  $C_1$  (e.g., *home\_address*) is endowed with a property  $R$  (e.g., *HasNecessarily*) linking it with a concept  $C_2$  (e.g., *postal\_code*). This situation can be formalized as:

$$\forall x(C_1(x) \rightarrow \exists y(C_2(y) \wedge R(x, y))); \quad (4)$$

meaning then, according to our example, that every home address is endowed with the property of having a postal code. As already stated in the previous section, properties can be systematically inherited along an inheritance hierarchy only under the “strict inheritance” hypothesis; instances inherit from the father concept.

The properties of  $C_i$ —i.e., the relationships between  $C_i$  and some other concepts of the ontology that define the classes of fillers for the slots of the frame associated with  $C_i$ —can pertain to several, different classes. According, e.g., to the classification used for the strictly “ontological” part of the NKRL language (see the next section and Zarri, 1997), these properties (slots) can be grouped in three different classes: “relations,” “attributes,” and “procedures” (see Table 1, where OID [object identifier] stands for the symbolic name of the particular concept or individual [instance of a concept]).

The slots of the “relation” type are used to represent mutual kinds of relationships between a concept or individual and other concepts or individuals. They can be “standard” or “user defined.” In Table 1, eight of them (“standard”) are shown; they are: *IsA* and the inverse *HasSpecialization*, *InstanceOf* and the inverse *HasInstance*, *MemberOf* (*HasMember*), and *PartOf*

(*HasPart*). Some of the formal properties of the relation properties are described in Table 2.

An important point concerns the fact that, because of the definitions of “concept” and “instance” given in the previous section and of the properties of *IsA*, *InstanceOf*, *PartOf*, and *MemberOf* illustrated in Table 2, a concept or an individual (instance) cannot make use of the totality of the eight “relations” introduced above. More exactly:

- The relation *IsA* and the inverse *HasSpecialization* are reserved to concepts.
- *HasInstance* can only be associated with a concept; *InstanceOf* with an individual (i.e., the concepts and their instances, the individuals, are linked by the *InstanceOf* and *HasInstance* relations).
- Moreover, *MemberOf* (*HasMember*) and *PartOf* (*HasPart*) can only be used to link concepts with concepts or instances with instances, but not concepts with instances; see also Winston, Chaffin, and Herrmann (1987).

The characteristic properties of a concept/individual are specifically represented by the slots of the “attribute” type. For example, for a concept like *tax\_*, possible attributes are *TypeOfFiscalSystem*, *CategoryOfTax*, *Territoriality*, *TypeOfTaxPayer*, *TaxationModalities*, etc. The restrictions about the sets of legal fillers for the “attribute” slots can be expressed using single concepts or particular combinations of concepts: e.g., an expression like “(INTERSECTION *human\_being* (UNION *doctor\_lawyer\_*) (NOT.ONE.OF *fred\_*))” can be used to designate a class of fillers that are men, can be doctors or lawyers, but cannot be Fred.

Figure 4 (an “ontology”) reproduces a fragment of Figure 1 (a “taxonomy”), where the concepts are now associated with their (highly schematized) defining frames; note that the two fillers (non-sortal concepts) *male\_ / female\_* could also have been replaced by their subsuming concept *sex\_*. This figure makes explicit what “inheritance of properties/attributes” means: Supposing that the frame for *mammal\_* is already defined and supposing now to tell the system that the concept *dog\_* is characterized by the two specific properties *Progeny* and *SoundEmission*, what the frame *dog\_* really includes is represented in the lower part of Figure 4.

The function assigned to slots of the “procedure” type is that of providing, in case, alternative inference and representation schemes in addition to the inheritance-based methods and representations. They normally supply various ways of attaching to frames “procedural” information, expressed using ordinary programming languages like Java, LISP, or C. A more detailed discussion in this context and an example can be found, e.g., in Bertino et al. (2001, pp. 152-154).

Table 1. Types of slots in a frame

<b>{ OID</b>	
<b>[ Relation</b>	( <i>IsA</i>   <i>InstanceOf</i> : <i>HasSpecialization</i>   <i>HasInstance</i> : <i>MemberOf</i>   <i>HasMember</i> : <i>PartOf</i>   <i>HasPart</i> :) ( <i>UserDefined</i> <sub>1</sub> : ... <i>UserDefined</i> <sub>n</sub> :)
<b>Attribute</b>	( <i>Attribute</i> <sub>1</sub> : ... <i>Attribute</i> <sub>n</sub> :)
<b>Procedure</b>	( <i>Procedure</i> <sub>1</sub> : ... <i>Procedure</i> <sub>n</sub> : ) }



Table 2. Some properties of *IsA*, *Instance Of*, *PartOf*, and *MemberOf*

$(IsA\ A\ B) \wedge (IsA\ B\ A) \leftrightarrow A \equiv B$ $(IsA\ A\ B) \wedge (IsA\ B\ C) \rightarrow (IsA\ A\ C)$ ( <i>IsA</i> is a partial order relationship) $(IsA\ A\ B) \wedge (IsA\ A\ C) \rightarrow \exists D (IsA\ B\ D) \wedge (IsA\ C\ D)$ $(PartOf\ A\ B) \rightarrow \neg (PartOf\ B\ A)$ $(PartOf\ A\ B) \wedge (PartOf\ B\ C) \rightarrow (PartOf\ A\ C)$ $(IsA\ A\ B) \wedge (PartOf\ B\ C) \rightarrow (PartOf\ A\ C)$ $(IsA\ A\ B) \wedge (PartOf\ A\ C) \rightarrow (PartOf\ B\ C)$ $(IsA\ B\ C) \wedge (A\ PartOf\ A\ C) \rightarrow (PartOf\ A\ B)$ $(IsA\ A\ B) \wedge (MemberOf\ B\ C) \rightarrow (MemberOf\ A\ C)$ $(InstanceOf\ C\ A) \wedge (IsA\ A\ B) \rightarrow (InstanceOf\ C\ B)$ $(PartOf\ C\ D) \wedge (InstanceOf\ C\ A) \wedge (InstanceOf\ D\ B) \rightarrow (PartOf\ A\ B)$ $(PartOf\ A\ B) \wedge (InstanceOf\ D\ B) \rightarrow \exists C (InstanceOf\ C\ A) \wedge (PartOf\ C\ D)$
--

Even if, under the influence of the Semantic Web work (see the companion article in this encyclopedia “Using Semantic Web Tools for Ontologies Construction”), the “classic” frame paradigm as expounded above is moving towards more formalized and logic-based types of representation, nevertheless, this paradigm still constitutes the foundation for the setup of a large majority of knowledge bases all over the world. Often, therefore, a knowledge base is nothing more than a “big” ontology formed of concepts/individuals represented under the form of frames. Unfortunately, frames suffer from congenital problems of unpredictability linked mainly (but not exclusively) with the arbitrariness in the choice of the slots. This arbitrariness is only partially obviated in some of the tools devoted to facilitate the setup of large knowledge bases by the use of meta-structures intended to describe in a precise way the computational behavior of a given slot—and therefore to give, in a certain way, also a sort of “definition” of the slot. A well-known approach in this context consists in adding “facets” to the slots, where a facet represents an annotation describing some characteristics of the slots, like type restrictions on the values of the slot (value-type facets) and specifications of the exact number of possible values that the slot may take on (cardinality facets).

To circumvent these unpredictability problems and enforce interoperability in the construction of knowledge bases—by allowing, e.g., the reuse of existing knowledge components or the import/export of ontologies—an Open Knowledge Base Connectivity (OKBC) protocol was developed in the late '90s; see, e.g., Chaudhri, Farquhar, Fikes, Karp, and Rice, (1998) and Table 3. OKBC is an application programming interface (API) specifying the operations that can be used to access a knowledge base by an application program; to do this, OKBC must make some assumptions about the knowledge representations used in the accessible knowledge bases and thus define, eventually, a “knowledge model,” to which such knowledge bases must conform. The OKBC knowledge model is very general, in order to include the representational

features supported by a majority of frame-based knowledge systems, and concerns general directions about the representation of constants, frames, slots, facets, etc. It allows frame-based systems to define their own behavior for many aspects of the knowledge model, e.g., with respect to the definition of the default values for the slots.

The most well-known OKBC-compatible tool for the setup of knowledge bases making use of the frame model is Protégé-2000, developed for many years at the Medical Informatics Laboratory of Stanford University (in California, USA); see, e.g., Noy, Fergerson, and Musen (2000) and Noy et al. (2001). Protégé-2000 represents today a sort of standard in the ontological domain. A Protégé ontology consists of “classes,” i.e., the concepts proper to a given application domain; “slots,” describing the properties or the attributes of the classes/concepts; “facets,” describing the properties of the slots; and “axioms,” used

Figure 4. A simple example explaining the inheritance of properties/attributes

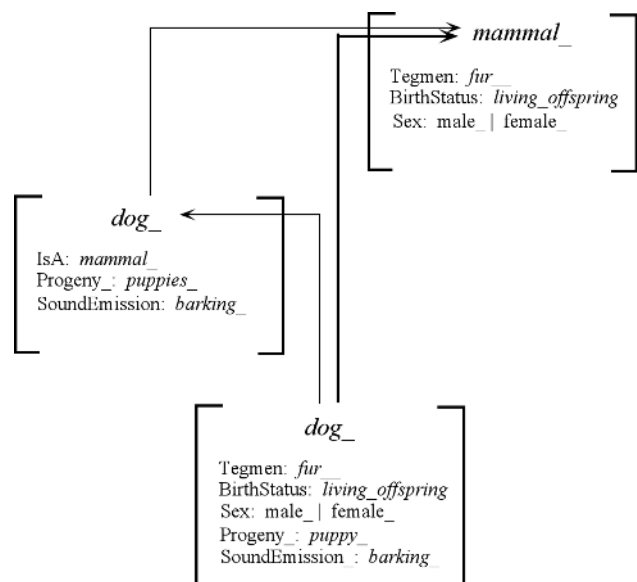
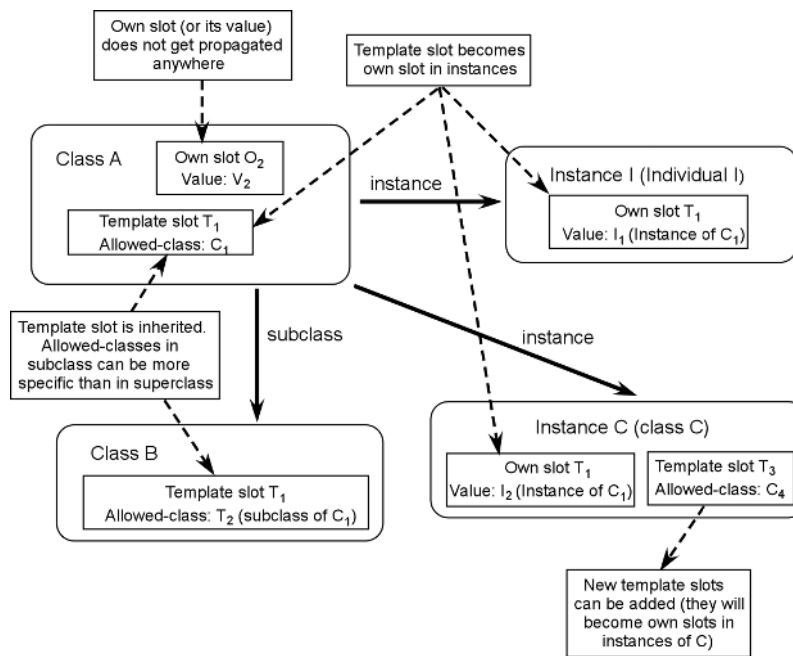


Figure 5. Propagation of template and own slots in Protégé-2000



to represent additional constraints. A Protégé “knowledge base” includes the ontology and the different “individuals” obtained as instances of classes when some specific values have been defined for the slots.

Definition of classes is a standard one: If a class  $B$  is a subclass of a class  $A$ , then every instance of  $B$  is also an instance of  $A$ . Multiple inheritance is admitted. A more controversial decision concerns the fact of admitting that both individuals and classes themselves can be instances of classes; see also Figure 5. In this way, Protégé can introduce “metaclasses” as classes whose instances are themselves classes. Every class (concept) has then a dual identity; it is a “subclass” of another class (its “superclass”) in the normal class hierarchy, and it is at the same time an “instance” of another class (its “metaclass”). As a class defines a sort of “compelling mold” for its instances, the metaclass defines a compelling mold for the associated classes, describing, e.g., which “own slots” (see below) these last can have and the constraints for the values of these slots. Metaclasses can be user-defined, allowing then the users to customize the Protégé-2000 tool according to the requirements of their domain and tasks.

The most important contribution of OKBC and Protégé to the development of “modern” frame systems concerns, however, the definition of the slots. Slots in Protégé and OKBC are frames in themselves; more importantly, they are first-class objects *that are defined independently of*

*any class*, i.e., independently from the requirements of any particular concept. Bearing in mind that slots represent properties or attributes of a concept, it is now evident that properties like Age or Name can be used to describe the characteristics of both (among many other things) the concepts *author\_* and *manuscript\_*. These slots will then be “attached,” when necessary, to *author\_*, *manuscript\_*, and many other possible concepts (classes). When a slot is “attached” to a class, it can take a “value.” In Protégé (and OKBC), the constraints that specify the allowed slot values are expressed through “facets.”

Slots can be attached to classes or instances in two ways, i.e., as “template slots” or as “own slots.” Own slots can be attached to both classes and instances; they define “local” properties of classes and instances, and, when attached to a class, they cannot be inherited by its subclasses and instances. Template slots can be attached only to classes; they can be inherited by the subclasses, and, when inherited by the instances of a class, they become own slots for these instances—analogously, own slots associated with classes were, originally, template slots of their specific metaclasses. The whole mechanism of the “attachment” is summarized in Figure 5, adapted from Figure 2 of Noy et al. (2000); see this last paper for more details on the attachment mechanism and for concrete examples. Table 3, derived from the same paper, summarizes the differences between the OKBC and Protégé-2000 knowledge models.

Table 3. Comparison between the OKBC (first column) and Protégé-2000 (second column) knowledge models

1 A frame (class or instance) can be an instance of multiple classes.	A frame (class or instance) can be an instance of only one class.
2 A frame (class or instance) has not to be necessarily an instance of a class.	A frame (class or instance) is always a instance of a class.
3 An own slot can be attached directly to any frame (class or instance).	An own slot attached to a frame is always derived from a template slot.
4 Classes, slots, facets, and individuals must not be necessarily frames.	Every class, slot, facet, and individual is necessarily a frame.
5 A frame can be at the same time a class, a slot, a facet, and an individual.	A frame is either a class, or a slot, or a facet, or an individual.

### FUTURE TRENDS: BEYOND “TRADITIONAL” ONTOLOGIES

A big amount of important, “economically relevant” information is buried into “narrative” information resources: This is true, e.g., for most of the corporate knowledge documents (memos, policy statements, reports, minutes, etc.), news stories, normative and legal texts, medical records, many intelligence messages, etc., as well as, in general, for a huge fraction of the information stored on the Web. In these narrative “documents,” the main part of the information content consists in the description of “events” that relate the real or intended behavior of some “actors” (characters, personages, etc.)—the term “event” is taken here in its more general meaning, covering also strictly related notions like fact, action, state, situation, etc. These actors try to attain a specific result, experience particular situations, manipulate some (concrete or abstract) materials, send or receive messages, buy, sell, deliver, etc. Note that, in these narrative documents, the actors or personages are not necessarily human beings; we can have narrative documents concerning, e.g., the vicissitudes in the journey of a nuclear submarine (the “actor,” “subject,” or “personage”) or the various avatars in the life of a commercial product. Note also that, even if a large amount of narrative documents concerns natural language (NL) texts, this is not necessarily true. A photo representing a situation that, verbalized, could be expressed as “Three nice girls are lying on the beach” is not of course an NL text, yet it is still a narrative document.

Because of the ubiquity of these “narrative” resources, being able to represent in a general, accurate, and effective way their semantic content—i.e., their key “meaning”—is then both conceptually relevant and economically important.

“Traditional” ontologies are not very suitable to deal with this problem. Their form of hierarchical organization is, in fact, largely sufficient to provide a *static, a priori* definition of the class/concepts and of their properties. This is no more true when we consider the *dynamic*

*behavior* of the concepts, i.e., we want to describe their mutual relationships when they take part in some concrete action, situation, etc. (“events”), in particular, when those “connectivity phenomena,” like causality, goal, indirect speech, coordination and subordination, etc., that link together “elementary events” must be taken into account. It is very likely, in fact, that, dealing with the sale of a company, the global information to represent is something like: “Company X has sold its subsidiary Y to Z *because* the profits of Y have fallen dangerously these last years *due to* a lack of investments,” or that, dealing with the relationships between companies in the biotechnology domain, “X made a milestone payment to Y *because* they decided to pursue an *in vivo* evaluation of the candidate compound identified by X.” It is now easy to imagine the awkward proliferation of totally *ad hoc* slots/properties that, sticking to a strict frame context, it would be necessary to introduce in order to approximate the connectivity phenomena in the above examples.

NKRL (Narrative Knowledge Representation Language; Zarri, 2003) innovates then with respect to the usual ontology paradigm by associating with the traditional ontologies of concepts, an “ontology of events,” i.e., a new sort of hierarchical organization where the nodes correspond to *n*-ary structures called “templates.”

Instead of using the traditional *object (class, concept) – attribute – value* organization, templates are generated from the combination of quadruples, connecting together the *symbolic name* of the template, a *predicate*, and the *arguments* of the predicate introduced by named relations, the *roles*. The quadruples have in common the “name” and “predicate” components. If we denote then with  $L_i$  the generic symbolic label identifying a given template, with  $P_j$  the predicate used in the template, with  $R_k$  the generic role, and with  $a_k$  the corresponding argument, the NKRL core data structure for templates has the following general format:

$$(L_i (P_j (R_1 a_1) (R_2 a_2) \dots (R_n a_n))) \tag{5}$$

See the example in Table 4. Predicates pertain to the set {BEHAVE, EXIST, EXPERIENCE, MOVE, OWN, PRO-

DUCE, RECEIVE}, and roles to the set {SUBJ(ect), OBJ(ect), SOURCE, BEN(e)F(iciary), MODAL(ity), TOPIC, CONTEXT}. An argument of the predicate can consist of a simple “concept” (according to the traditional “ontological” meaning of this word) or of a structured association (“expansion”) of several concepts.

Templates are included in turn in an inheritance hierarchy, HTemp(lates), which implements the new “ontology of events”; they represent then formally generic classes of elementary events like “move a physical object,” “be present in a place,” “produce a service,” “send/receive a message,” “build up an Internet site,” etc. When a particular event pertaining to one of these general classes must be represented, the corresponding template is “instantiated” to produce what, in the NKRL’s jargon, is called a “predicative occurrence.”

To represent then a narrative event like “British Telecom will offer its customers a pay-as-you-go (payg) Internet service in autumn 1998,” we must select firstly in the HTemp hierarchy the template corresponding to “supply a service to someone,” represented in the upper part of Table 4. This template is a specialization (see the “father” code) of the particular MOVE template of HTemp corresponding to “transfer of resources to someone.” In a template, the arguments of the predicate (the  $a_k$  terms in (5)) are represented by variables with associated constraints—which are expressed as concepts or combinations of concepts, i.e., using the terms of the NKRL standard “ontology of concepts” (HClass, “hierarchy of classes”). The constituents (as Source in Table 4) included in square brackets are optional. When deriving a predicative occurrence (an instance of a template) like c1 in Table 4, the role fillers in this occurrence must conform

to the constraints of the father template. For example, in occurrence c1, british\_telecom is an individual instance of the concept *company\_*, which is, in turn, a specialization of *human\_being\_or\_social\_body*; *payg\_internet\_service* is a specialization of *service\_*, a specific term of *social\_activity*, etc. The meaning of the expression “BENF (SPECIF *customer\_british\_telecom*)” in c1 is self-evident: The beneficiaries (role BENF) of the service are the customers of—SPECIF(ication)—British Telecom. In the occurrences, the two operators date-1 and date-2 materialize the temporal interval normally associated with narrative events; see Zarri (1998).

About 180 templates are permanently inserted into Htemp; this ontology of events corresponds then to a sort of “catalogue” of narrative formal structures that, moreover, are very easy to “customize” in order to derive the new templates that could be needed for a particular application. This approach is particularly advantageous for practical applications, and it implies, in particular, that: (i) a system builder does not have to create himself the structural knowledge needed to describe the events proper to a (sufficiently) large class of narrative documents; and (ii) it becomes easier to secure the reproduction or the sharing of previous results.

The basic NKRL knowledge representation tools are complemented by more complex mechanisms intended to represent, among other things, those “connectivity phenomena” already mentioned that, in a sequence of statements, cause the global meaning to go beyond the simple addition of the information conveyed by each single statement. This is obtained making use of second-order structures created through *reification* of the predicative occurrences’ conceptual labels. For example, the “bind-

Table 4. Deriving a predicative occurrence from a template

```

name: Move:TransferOfService
father: Move:TransferToSomeone
position: 4.24
NL description: 'Transfer or Supply a Service to Someone'

MOVE   SUBJ      var1: [var2]
        OBJ       var3
        [SOURCE   var4: [var5]]
        BENF     var6: [var7]
        [MODAL   var8]
        [TOPIC   var9]
        [CONTEXT var10]
        {[modulators]}

var1 = <human_being_or_social_body>
var3 = <service_>
var4 = <human_being_or_social_body>
var6 = <human_being_or_social_body>
var8 = <process_> <sector_specific_activity>
var9 = <sortal_concept>
var10 = <situation_>
var2, var5, var7 = <geographical_location>

c1)    MOVE SUBJ      british_telecom
        OBJ       payg_internet_service
        BENF     (SPECIF customer_british_telecom)
        date-1:  after-1-september-1998
        date-2:
    
```



ing occurrences” are NKRL structures consisting of lists of symbolic labels ( $c_i$ ) of predicative occurrences; the lists are differentiated making use of specific binding operators like GOAL and CAUSE. If, in Table 4, we would state that “British Telecom *intends* to offer to its customers a pay-as-you-go (payg) Internet service...,” we should introduce an additional predicative occurrence simplified here as:  $c_2$ ) BEHAVE SUBJ british\_telecom, and meaning that “at the date (date-1) associated with  $c_2$ , it can be noticed that British Telecom is acting in some way.” We will then add a *binding occurrence*  $c_3$  of the form:  $c_3$ ) (GOAL  $c_2$   $c_1$ ). The meaning of  $c_3$  is that: “the activity described in  $c_4$  is focalized towards (GOAL) the realization of  $c_1$ ,” where  $c_1$  is represented in Table 4.

Reasoning ranges from the direct questioning of an NKRL knowledge base making use of “search patterns” (the formal NKRL equivalents of natural language queries) that try to unify the predicative occurrences of the base, to high-level inference procedures employing a complex inference engine. For example, the “transformation rules” try to “adapt,” from a semantic point of view, the original query/queries (search patterns) that failed to the real contents of the system knowledge base. The principle employed consists in using rules to automatically “transform” the original query (i.e., the original search pattern) into one or more different queries (search patterns) that are not strictly “equivalent” but only “semantically close” to the original one. In this way, an original query posed, e.g., in terms of “searching for evidence of having lived in a given country” will be replied in terms of “searching for evidence of an original school/university diploma delivered in that country.” “Hypothesis rules” allow building up “reasonable” answers according to a number of predefined reasoning schemata, e.g., “causal” schemata. For example, after having directly retrieved information like: “Pharmacopeia, an USA biotechnology company, has received \$64,000,000 from the German company Schering in connection with an R&D activity,” we could be able to automatically construct a sort of “causal explanation” of this information by retrieving in the knowledge base information like: (i) “Pharmacopeia and Schering have signed an agreement concerning the production by Pharmacopeia of a new compound,” and (ii) “In the framework of the agreement previously mentioned, Pharmacopeia has actually produced the new compound.”

Other systems exist that, being based on knowledge representation schemata richer than those of the traditional ontologies, can assure, as in NKRL, a level of inferencing neatly more “appealing,” powerful, and concise than that supplied by “traditional” ontological systems. Conceptual graphs (CGs) are a representation system developed by John Sowa (1999) and derived from

early work in the Semantic networks domain that makes use of a graph-based notation for representing “concepts” (and “concept referents”) and “conceptual relations.” Like NKRL, CGs can be used to represent in a formal way narrative events like “A pretty lady is dancing gracefully” and more complex, second-order constructions like contexts, wishes, and beliefs. CYC (see Lenat, Guha, Pittman, Pratt, & Shepherd, 1990) concerns one of the most controversial endeavors in the history of artificial intelligence. Started in the early ’80s as a MCC (Microelectronics and Computer Technology Corp., Texas, USA) project, it ended about 15 years later with the setup of an enormous knowledge base containing about a million hand-entered “logical assertions,” including both simple statements of facts and rules about what conclusions can be inferred if certain statements of facts are satisfied. The “upper level” of the ontology that structures the CYC knowledge base is now freely accessible on the Web; see <http://www.cyc.com/cyc/opencyc>. A detailed analysis of the origins, developments, and motivations of the CYC project can be found in Bertino et al. (2001, pp. 275-316).

## CONCLUSION

From the beginning of the ’70s, artificial intelligence (AI) and databases have shared some common interests. The origin of this convergence can be traced back, in a very concise way, to the lack of *semantic support* that characterized the “traditional” DBs, while early AI systems were limited in their capacity to supply and maintain *large amounts of factual data*. The so-called intelligent database systems (IDBSs; Bertino et al., 2001) are the result of this mutual impregnation—and ontologies are now presented as the ultimate instrument for the setup of “true” IDBSs. We think that this assertion is not totally unreasonable on the condition that the word “ontology” could be understood according to its widest meaning—to include also, e.g., the dynamic behavior and the mutual relationships among concepts and not only some static, a priori definition of these concepts and of their properties. As we have seen in the previous section, this can be realized by adding, as in NKRL, “ontologies of events” to the traditional “ontologies of concepts.”

## REFERENCES

Bertino, E., Catania, B., & Zarri, G. P. (2001). *Intelligent database systems*. London: Addison-Wesley and ACM Press.



Brachman, R. J. (1983). What IS-A is and isn't: An analysis of taxonomic links in semantic networks. *IEEE Computer*, 16(10), 30-36.

Brachman, R. J. (1985). "I lied about the trees," or, defaults and definitions in knowledge representation. *AI Magazine*, 6(3), 80-93.

Chaudhri, V. K., Farquhar, A., Fikes, R., Karp, P. D., & Rice, J. P. (1998). OKBC: A programmatic foundation for knowledge base interoperability. In *Proceedings of the 1998 National Conference on Artificial Intelligence, AAAI/98*. Cambridge, MA: MIT Press/AAAI Press.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5, 199-220.

Guarino, N., & Giaretta, P. (1995). Ontologies and knowledge bases: Towards a terminological clarification. In N.J.I. Mars (Ed.), *Towards very large knowledge bases: Knowledge building & knowledge sharing*. Amsterdam: IOS Press.

Lehmann, F. (Ed.). (1992). *Semantic networks in artificial intelligence*. Oxford, UK: Pergamon Press.

Lenat, D. B., Guha, R. V., Pittman, K., Pratt, D., & Shepherd, M. (1990) CYC: Toward programs with common sense. *Communications of the ACM*, 33(8), 30-49.

Noy, N. F., Fergerson, R. W., & Musen, M. A. (2000). The knowledge model of Protégé-2000: Combining interoperability and flexibility. In R. Dieng & O. Corby, O. (Eds.), *Knowledge Acquisition, Modeling, and Management: Proceedings of the European Knowledge Acquisition Conference, EKAW'2000*. Berlin, Germany: Springer-Verlag.

Noy, N. F., Sintek, M., Decker, S., Crubezy, M., Fergerson, R. W., & Musen, M. A. (2001). Creating Semantic Web contents with Protégé-2000. *IEEE Intelligent Systems*, 16(2), 60-71.

Reiter, R. (1980). A logic for default reasoning. *Artificial Intelligence*, 13, 81-132.

Sowa, J. F. (1999). *Knowledge representation: Logical, philosophical, and computational foundations*. Pacific Grove, CA: Brooks Cole.

Touretzky, D. S. (1986). *The mathematics of inheritance systems*. London: Pitman.

Winston, M. E., Chaffin, R., & Herrmann, D. (1987). A taxonomy of part-whole relations. *Cognitive Science*, 11, 417-444.

Zarri, G. P. (1997). NKRL, a knowledge representation tool for encoding the "meaning" of complex narrative texts. *Natural Language Engineering*, 3, 231-253.

Zarri, G. P. (1998). Representation of temporal knowledge in events: The formalism, and its potential for legal narratives. *Information & Communications Technology Law*, 7, 213-241.

Zarri, G. P. (2003). A conceptual model for representing narratives. In R. Jain, A. Abraham, C. Faucher, & van der Zwaag (Eds.), *Innovations in knowledge engineering*. Adelaide, South Australia, Australia: Advanced Knowledge.

## KEY TERMS

**Frame-Based Representation:** A way of defining the "meaning" of a concept by using a set of properties ("frame") with associated classes of admitted values—this "frame" is linked with the node representing the concept. Associating a frame with the concept  $c_i$  to be defined corresponds to establishing a *relationship* between  $c_i$  and *some* of the other concepts of the ontology; this relationship indicates that the concepts  $c_1, c_2 \dots c_n$  used in the frame defining  $c_i$  denote the "class of fillers" (specific concepts or instances) that can be associated with the "slots" (properties, attributes, qualities, etc.) of the frame for  $c_i$ .

**Inheritance Hierarchies:** Ontologies/taxonomies are structured as hierarchies of concepts ("inheritance hierarchies") by means of "IsA" links. A semantic interpretation of this relationship among concepts, when noted as (IsA  $B A$ ), means that concept  $B$  is a *specialization* of the more general concept  $A$ . In other terms,  $A$  subsumes  $B$ . This assertion can be expressed in logical form as:

$$\forall x (B(x) \rightarrow A(x)) \tag{1}$$

(1) says that, e.g., if any *elephant\_* ( $B$ , a concept) IsA *mammal\_* ( $A$ , a more general concept), and if *clyde\_* (an instance or individual) is an *elephant\_*, then *clyde\_* is also a *mammal\_*.

**IsA and InstanceOf Links:** The necessary complement of IsA for the construction of well-formed hierarchies is the InstanceOf link, used to introduce the "instances" (concrete examples) of the general notions represented by the concepts. The difference between (IsA  $B A$ ) and (InstanceOf  $C B$ ) can be explained by considering  $B$  as a subclass of  $A$  in the IsA case, operator " $\subset$ ," and by

considering  $C$  as a member of the class  $B$  in the InstanceOf case, operator “ $\in$ .” The notion of instance is, however, much more controversial than that of concept. Problems about the definition of instances concern, e.g., (i) the possibility of accepting that concepts (to the exclusion of the root) could also be considered as “instances” of their generic concepts and (ii) the possibility of admitting several levels of instances, i.e., instances of an instance.

**NKRL:** The Narrative Knowledge Representation Language. “Classical” ontologies are largely sufficient to provide a *static, a priori* definition of the concepts and of their properties. This is no more true when we consider the *dynamic behavior* of the concepts, i.e., we want to describe their mutual relationships when they take part in some concrete action, situation, etc. (“events”). NKRL deals with this problem by adding to the usual ontology of concept an “ontology of events,” a new sort of hierarchical organization where the nodes, called “templates,” represent general classes of events like “move a physical object,” “be present in a place,” “produce a service,” “send/receive a message,” etc.

**Ontologies vs. Taxonomies:** In a taxonomy (and in the most simple types of ontologies), the *implicit* definition of a concept derives simply by the fact of being inserted in a network of *specific/generic relationships* (IsA) with the other concepts of the hierarchy. In a “real” ontology, we must supply also some *explicit* definitions for the concepts—or at least for a majority among them. This can be obtained, e.g., by associating a “frame” (see above) with these concepts.

**Open Knowledge Base Connectivity (OKBC) Protocol:** A protocol aiming at enforcing interoperability in the construction of knowledge bases. The OKBC

knowledge model is very general, in order to include the representational features supported by a majority of frame-based knowledge systems, and concerns general directions about the representation of constants, frames, slots, facets, etc. It allows frame-based systems to define their own behavior for many aspects of the knowledge model, e.g., with respect to the definition of the default values for the slots. The most well-known OKBC-compatible tool for the setup of knowledge bases making use of the frame model is Protégé-2000, developed for many years at the Medical Informatics Laboratory of Stanford University, and that represents today a sort of standard in the ontological domain.

**Overriding and Multiple Inheritance:** Two among the main theoretical problems that can affect the construction of well-formed inheritance hierarchies. Overriding (or “defeasible inheritance,” or “inheritance with exceptions”) consists in the possibility of admitting exceptions to the “strict inheritance” interpretation of an IsA hierarchy. Under the complete overriding hypothesis, the values associated with the different properties of the concepts, and the properties themselves, must be interpreted simply as “defaults,” that is, always possible to modify. An unlimited possibility of overriding can give rise to problems of logical incoherence; Reiter’s default logic has been proposed to provide a formal semantics for inheritance hierarchies with defaults. In a “multiple inheritance” situation, a concept can have multiple parents and can inherit properties along multiple paths. When overriding and multiple inheritance combine, we can be confronted with very tricky situations like that illustrated by the well-known “Nixon diamond.”



# Ontology-Based Data Integration

**Agustina Buccella**

*Universidad Nacional del Comahue, Argentina*

**Alejandra Cechich**

*Universidad Nacional del Comahue, Argentina*

**Nieves Rodríguez Brisaboa**

*Universidade da Computação, Spain*

## INTRODUCTION

Nowadays, different areas of large modern enterprises use different database management systems to store and search their critical data. Competition, evolving technology, geographical distribution, and the inevitable growing decentralization contribute to this diversity. All of these databases are very important to an enterprise, but their different interfaces make their administration difficult. Therefore, recovering information through a common interface becomes crucial to realize, for instance, the full value of data contained in the databases (Hass & Lin, 2002).

In the 1990s, the term *federated database* emerged to characterize techniques for proving an integrating data access, giving a set of distributed, heterogeneous, and autonomous databases (Busse, Kutsche, Leser & Weber, 1999; Litwin, Mark, & Roussoupoulos, 1990; Sheth & Larson, 1990). Even these concepts are rather well-known today, a brief introduction would clarify their meanings in the context of our paper. They are as follows:

- **Autonomy:** Users and applications can access data through a federated system or by their own local system. Autonomy can be classified into three types: *design autonomy*, *communication autonomy* and *execution autonomy* (Busse et al., 1999; Ozsu & Valduriez, 1999). The first type refers to the data model independence, the second type involves the different ways of communication among the systems, and the third type refers to the independence of the execution of the local operations.
- **Distribution:** In the last years and with the arrival of the Internet it is very common to see computers connected by some type of network. Generally speaking, data may be distributed among multiple sources and stored in a single computer system or

in multiple computer systems. These computer systems may be geographically distributed but interconnected by a communication network.

- **Heterogeneity:** Different meanings that may be inferred from data stored in databases. In Cui and O'Brien (2000), heterogeneity is classified into four categories: *structural*, *syntactical*, *system*, and *semantic*. Structural heterogeneity deals with inconsistencies produced by different data models; syntactical heterogeneity deals with consequences of using different languages and data representations; system heterogeneity deals with having different supporting hardware and operating systems; and semantic heterogeneity is further classified as (a) dealing with semantically equivalent concepts; some models use different terms to refer to the same concept (e.g., synonyms, or properties, are modeled differently by different systems); (b) dealing with semantically unrelated concepts; the same term may be used by different systems to denote completely different concepts; and (c) dealing with semantically related concepts by using generalization/specification, different classifications, and so forth. Additionally, a similar classification of heterogeneity can be found in Goh (1996).

In this paper, we will focus on the use of ontologies because of their advantages when using them for data integration. For example, an ontology may provide a rich, predefined vocabulary that serves as a stable conceptual interface to the databases and is independent of the database schemas; knowledge represented by the ontology may be sufficiently comprehensive to support translation of all relevant information sources; and an ontology may support consistency management and recognition of inconsistent data. The next section will analyze several systems using ontologies as a tool to solve data integration problems.

## BACKGROUND

Recently, the term *federated databases* has evolved into *federated information systems* (FIS) because of the diversity of new information sources involved in the federation, such as HTML pages, databases, and filing, either static or dynamic.

A useful classification of information systems based on the dimensions of distribution and heterogeneity can be found in Busse et al., 1999). Figure 1 shows this classification where the addition of an additional dimension—autonomy—makes the term *Federated Information Systems* appears.

Besides, the work reported in Busse et al., 1999 defines the classical architecture of federated systems (based on Sheth & Larson [1990]), which is widely referred by many researches. Figure 2 shows this architecture. In the figure, the *wrapper layer* involves a number of modules belonging to a specific data organization. These modules know how to retrieve data from the underlying sources hiding their data organizations. As the federated system is autonomous, local users may access local databases through their local applications

Figure 1. Classification of information systems.

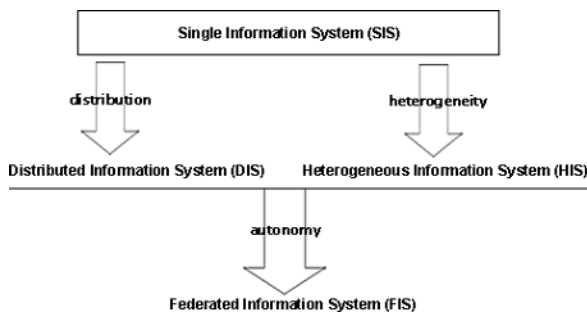
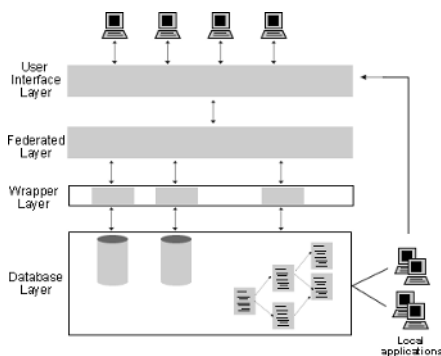


Figure 2. Architecture of Federated Systems



independently from users of other systems. Otherwise, to access the federated system, they need to use the *user interface layer*.

The *federated layer* is one of the main components currently under analysis and study. Its importance comes from its responsibility to solve the problems related to the semantic heterogeneity, as was previously introduced. So far, different approaches have been used to model this layer. They are as diverse as complementary in some cases, and can involve different perspectives such as the use of ontologies (Ambite et al., 1997; Buccella, Cechich, & Brisaboa, 2003; Goh, Bressan, Siegel, & Madnick, 1999; Gray et al., 1997), the use of metadata (Busse et al., 1999; Nam & Wang, 2002; Seligman & Rosenthal, 1996).

## ONTOLOGY-BASED DATA INTEGRATION

The term *ontology* was introduced by Gruber (1993) as an “*explicit specification of a conceptualization.*” A *conceptualization*, in this definition, refers to an abstract model of how people commonly think about a real thing in the world; and *explicit specification* means that concepts and relationships of the abstract model receive explicit names and definitions.

An ontology gives the name and description of the domain-specific entities by using predicates that represent relationships between these entities. The ontology provides a vocabulary to represent and communicate domain knowledge along with a set of relationships containing the vocabulary’s terms at a conceptual level. Therefore, because of its potential to describe the semantic of information sources and to solve the heterogeneity problems, the ontologies are being used for data integration tasks.

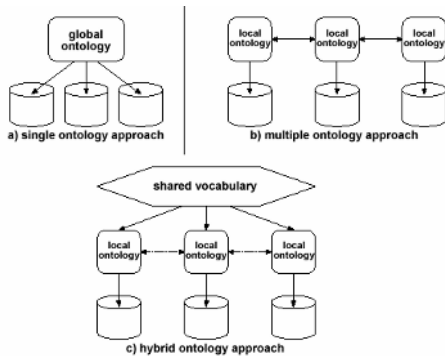
Some surveys on ontology-based systems for data integration can be found in literature. For example, Wache et al. (2001) focused on some aspects of the use of ontologies: the language representation, mappings, and tools. This work also classified the use of ontologies into three approaches: *single ontology approach*, *multiple ontology approach*, and *hybrid ontology approach* (see Figure 3).

Another different survey comparing the expressiveness of the languages can be found in Corcho and Gomez-Perez (2000), but only languages to represent ontologies are compared in this case.

This section will focus on how ontology-based systems address the semantic heterogeneity problems. We have investigated many systems, which follow some of the approaches of Figure 3, considering the relevant aspects shown in the hierarchy of Figure 4. This hierar-



Figure 3. Classification of the use of ontologies



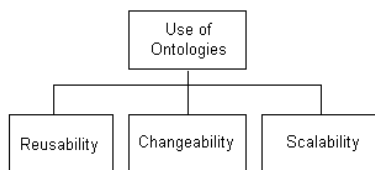
chy does not mean specialization, rather whether quality properties are improved due a particular use of ontologies in a given system.

The *use of ontologies* refers to how ontologies help solve data integration problems. Commonly, the systems show how ontologies are used to solve the integration problems and how ontologies interact with other architectural components. In this aspect the systems describe their ontological components and the ways to solve the different semantic heterogeneity problems.

Then we characterize this feature by means of three quality characteristics: *reusability*, *changeability*, and *scalability*. *Reusability* refers to the ability of reusing the ontologies, that is, ontologies defined to solve other problems can be used in a system because (a) the system supports different ontological languages and (b) the system defines local ontologies. *Changeability* refers to the ability of changing structures within an information source without producing substantial changes in the system components. Finally, *scalability* refers to the possibility of easily adding new information sources to the integrated system.

Following, we will describe how some ontology-based systems implement the aspects of Figure 4. In general, we have chosen systems widely referred by researchers, although some of them are still under development. Additionally, we have analyzed other recent systems that implement new ideas to solve heterogeneity problems.

Figure 4. Main aspects of the use of ontologies



Firstly, with respect to the *use of the ontologies*, we can find similarities among some systems. For example the ontologies defined in SIMS (Ambite et al., 1997; Arens, Hsu, & Knoblock, 1993, 1996) and Carnot (Woelk, Cannata, Huhns, Shen, & Tomlinson, 1993) do not facilitate reusability because both define a global ontology (single ontology approach) in order to integrate data. The same is true with changeability because no support is given by these systems to bear changes in the information sources. When one source changes, the global ontology must be rebuilt. Finally, scalability is not supported at all because, again, adding new information sources will generate the rebuilding of the global ontology in order to include the new terms and mappings not considered yet.

In general, the problem of dealing with a global ontology approach is that we must manage a global integrated ontology, which involves administration, maintenance, consistency, and efficiency problems that are very hard to solve. For example, SIMS requires constructing a general domain model that encompasses the relevant parts of the database schemas. Because each database model is related to this general domain model, the integration problem is shifted from how to build a single integrated model to how to map the domain and the information source models. Other similar examples are the MediWeb (Arruda, Baptista, & Lima, 2002) and DIA (Medcraft, Schiel, & Baptista, 2003) systems. On one hand, MediWeb uses XML documents as information sources (Baru, 1998). XML provides only a syntactical view of the underlying data. To allow for precise querying and semantic interpretation, the XML documents should be complemented with a conceptual model that adequately describes the semantics of its tags. The ontologies arise to fulfil this semantic gap, enabling a precise semantic expression for querying XML documents. These ontologies are used as a common schema. Therefore, the scalability and changeability are not very good, because a change in one source or the addition of a new source generates a modification in its global ontology. On the other hand, DIA does not provide any support to reach reusable ontologies. But the changeability and scalability can be improved, because this global ontology does not need to be modified when a source is added; only an ontology-schema matching table is necessary to add a new database to the integrated system.

Otherwise, systems such as OBSERVER (Mena, Kashyap, Sheth, & Illarramendi, 1996, 2000), use the multiple ontologies approach to alleviate the problems presented by the single approach. OBSERVER defines a model for dealing with multiple ontologies avoiding problems about integrating global ontologies. The different ontologies (*user ontologies*) can be described



using different vocabularies depending on the user's needs. In OBSERVER, every information source has only one *ontology server*, a module that provides information about ontologies located in the node as well as about their underlying data repositories. The system allows encapsulating any direct interaction with user ontologies and also any access to data repositories. Another important component of the OBSERVER system is its *IRM shared repository*. It can be seen as a catalog of semantics of the system used to solve the “*vocabulary problem*” (heterogeneous vocabularies used to describe the same information). The IRM component supports the ontology-based interoperation defining several kinds of interoperable relationships, such as synonym, hyponym, hypernym, and overlap, among the terms of different (locally developed) ontologies. The reusability is widely supported by OBSERVER because the local ontologies might be defined for other purposes.

From the perspective of the hybrid ontology approach, we can find that the KRAFT (Gray et al. 1997; Preece, Hui, & Gray, 1999; Preece, Hui, Gray, Jones, & Cui, 1999) system defines two kinds of ontologies: a *local ontology* and a *shared ontology*. For each knowledge source there is one local ontology and the shared ontology formally defines the terminology of the domain problem. In order to avoid the problems about semantics heterogeneity, that might occur between a local ontology and the shared ontology (ontology mismatches; Visser, Jones, Bench-Capon, & Shave, 1998), an *ontology mapping* is also defined for each knowledge source. It is a partial function that maps terms and expressions defined by a local ontology to terms and expressions of the shared ontology. As the local ontologies can be defined independently, the reusability is possible. Also, the changeability and the scalability are better supported because when a source change or a new source needs to be added, only the local ontologies and the mapping ontology must be modified by including the changed or new information.

Something similar happens with mediators over ontology-based information sources (MOIS; Tzitzikas, Spyrtatos, & Constantopoulos, 2001), COIN (Firat, Madnick, & Grosz, 2002; Goh et al., 1999; Siegel & Madnick, 1991) and InfoSleuth (Bayardo et al., 1997; Deschaine, Brice, & Nodine, 2000; Woelk & Tomlinson, 1994) In the case of MOIS, each source is associated to an ontology, which consists of a set of terms structured by a subsumption relation. Also, MOIS has a *mediator*, a secondary source that can bridge the heterogeneities between two or more sources and can provide a unified access to those sources. Besides, the mediator has a number of *articulations* to the sources. An articulation to a source is a set of relationships between the terms of

the mediator and the terms of that source. The system does not merge the ontologies to solve heterogeneity problems among the sources, instead of this, if the same term appears within two sources, both terms are not assumed as equivalent because they may have different interpretations (meanings). Two terms are considered equivalent only if they can be shown to be equivalent using the articulations of each source. The addition of a new source is a simple task because only two steps are necessary: the selection of a mediator in the network and the design of an articulation between the selected mediator and the new source.

In the case of COIN, it contains three main components: *context axioms*, *elevation axioms*, and a *domain model*. The domain model is a collection of the source's primitive types, such as strings or integers, and semantics types that define the application domain corresponding to the integrated data sources. The elevation axioms act as a mapping between attributes in the source and semantics types in the domain model. Finally, the context axioms define alternative interpretations of the semantics objects in different contexts. All of these components and the object oriented approach used by COIN, generate the domain model does not require to be updated every time a new source is added. The main advantage of COIN is allowing knowledge of data semantics to be independently captured in sources while let a specialized mediator (context mediator) detect and solve potential conflicts at the time a query is submitted.

As a final case, the InfoSleuth system is an agent-based system that provides interoperation among autonomous systems. To add a resource to the system the resource only needs to have an interface to advertise its services and let other agents make use of it immediately.

## FUTURE TRENDS

Several systems compared here are still in a development stage and, as we have explained, some problems should be solved in order to reach a good integration. For example, systems as SIMS and Carnot use a global ontology which decreases reusability, changeability and scalability. Other systems propose multiple or hybrid approaches to avoid these problems, but additional efforts have to be made to reach reusability.

Finally, automatic or semiautomatic tools are necessary to help the integration process—tools to map different ontologies or assistant-tools to create the ontologies. For example, the DOME (Cui, Jones, & O'Brien, 2001; Cui & O'Brien, 2000) system uses ontology extraction tools to generate the initial ontologies. Specifically, DOME uses XRA (Yang, Cui, & O'Brien, 1999), which is an ontology extraction tool

that uses a reverse engineering approach to extract an initial ontology from given data sources and their application programs.

## CONCLUSION

Today, the semantic heterogeneity involves many complex problems that are addressed by using different approaches, among them the use of ontologies, which give a higher degree of semantics to the treatment of data involved in the integration.

We have analyzed several systems accordingly with the use of the ontologies, and we have evaluated three important aspects also related with them—reusability, changeability, and scalability. Each system has implemented its own solution with advantages and disadvantages, but some elements in common and some original aspects can be found. There are other important aspects we could have considered for characterizing current ontology-based systems. Among them, we should mention the use of automated tools for supporting the manipulation of ontologies.

Of course, further characterization is needed to completely understand the use of ontologies in data integration. We hope our work will motivate the readers to deeper immerse themselves in this interesting world.

## REFERENCES

Ambite, J. L., Arens, Y., Ashish, N., Knoblock, C. A., et al. (1997, December 22). *The SIMS manual 2.0* (University of Southern California Tech. Rep.). Retrieved October 10, 2003, from <http://www.isi.edu/sims/papers/sims-manual.ps>

Arens, Y., Hsu, C., & Knoblock, C. A. (1993). Retrieving and integrating data from multiple information sources. *International Journal in Intelligent and Cooperative Information Systems*, 2(2), 127-158.

Arens, Y., Hsu, C., & Knoblock, C. A. (1996). Query processing in the SIMS information mediator. In A. Tate (Ed.), *Advanced Planning Technology* (pp. 61-69). Menlo Park, CA: AAAI Press.

Arruda, L., Baptista, C., & Lima, C. (2002). MEDIWEB: A mediator-based environment for data integration on the Web. *Databases and Information Systems Integration. ICEIS*, 34-41.

Baru C. (1998). Features and requirements for an XML view definition language: Lessons from XML information mediation. Paper presented at the *W3C Workshop on*

*Query Language (QL'98)*. Retrieved March 5, 2004, from <http://www.w3.org/TandS/QL/QL98/pp/xmas.html>

Bayardo, R. J., Jr., Bohrer, W., Brice, R., Cichocki, A., Fowler, J., Helal, A., et al. (1997). InfoSleuth: Agent-based semantic integration of information in open and dynamic environments. *Proceedings from the ACM SIGMOD International Conference on Management of Data* (pp. 195-206).

Buccella, A., Cechich, A., & Brisaboa, N. R. (2003). An ontology approach to data integration. *Journal of Computer Science and Technology*, 3(2), 62-68. Retrieved March 20, 2004, from <http://journal.info.unlp.edu.ar>

Busse, S., Kutsche, R. D., Leser, U., & Weber H. (1999). *Federated information systems: Concepts, terminology and architectures* (Tech. Rep. No. 99-9). Berlin, Germany: Technische Universität.

Corcho, O., & Gomez-Perez, A. (2000). Evaluating knowledge representation and reasoning capabilities of ontology specification languages. *Proceedings of the ECAI 2000 Workshop on Applications of Ontologies and Problem-Solving Methods*. Retrieved February 5, 2004, from <http://delicias.dia.fi.upm.es/WORKSHOP/ECAI00/schedule.html>

Cui, Z., Jones, D., & O'Brien, P. (2001). Issues in Ontology-based Information Integration. *Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing* (pp. 141-146).

Cui, Z., & O'Brien, P. (2000). Domain ontology management environment. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*.

Deschaine, L. M., Brice, R. S., & Nodine, M. (2000). *Use of InfoSleuth to coordinate information acquisition, tracking and analysis in complex applications* (Tech. Rep. No. MCC-INSL-008-00).

Firat, A., Madnick, S., & Grosz, B. (2002). Knowledge integration to overcome ontological heterogeneity: challenges from financial information systems. *Twenty-Third International Conference on Information Systems*.

Goh, C. H. (1996). *Representing and reasoning about semantic conflicts in heterogeneous information sources*. Doctoral dissertation, Massachusetts Institute of Technology, Sloan School of Management.

Goh, C. H., Bressan, S., Siegel, M., & Madnick, S. E. (1999). Context interchange: New features and formalisms for the intelligent integration of information. *ACM Transactions on Information Systems*, 17(3), 270-293.

Gray, P. M. D., Preece, A., Fiddian, N. J., et al. (1997). KRAFT: Knowledge fusion from distributed databases

and knowledge bases. *Proceedings of the 8th International Workshop on Database and Expert Systems Application* (pp. 682-691).

Gruber, T. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), 199-220.

Hass, L., & Lin, E. (2002). IBM federated database technology. Retrieved September 25, 2003, from <http://www-106.ibm.com/developerworks/db2/library/techarticle/0203haas/0203haas.html>

Litwin, W., Mark, L., & Roussoupoulos, N. (1990). Interoperability of multiple autonomous databases. *ACM Computing Surveys*, 22(3), 267-293.

Medcraft, P., Schiel, U., & Baptista, P. (2003). DIA: Data integration using agents. *Databases and Information Systems Integration. ICEIS*, 79-86.

Mena, E., Kashyap, V., Sheth, A., & Illarramendi, A. (1996). Managing multiple information sources through ontologies: relationship between vocabulary heterogeneity and loss of information. *Proceedings of Knowledge Representation Meets Databases, ECAI'96 Conference*, Budapest, Hungary (pp. 50-52).

Mena, E., Kashyap, V., Sheth, A., & Illarramendi, A. (2000). *Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies* (pp. 1-49). Boston: Kluwer Academic.. Retrieved February 15, 2004, from <http://citeseer.nj.nec.com/mena96observer.html>

Nam, Y., & Wang, A. (2002). Metadata Integration Assistant Generator for Heterogeneous Distributed Databases. *Proceedings of the International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems*, Irvine, CA (pp. 28-30).

Ozsu, M. T., & Valduriez, P. (1999). *Principles of distributed database systems* (2nd Ed.). Prentice Hall.

Preece, A., Hui, K., Gray, A., Jones, D., & Cui, Z. (1999). The KRAFT architecture for knowledge fusion and transformation. *Proceedings of the 19th SGES International Conference on Knowledge-Based Systems and Applied Artificial Intelligence*, Berlin, Germany. Retrieved February 15, 2004, from <http://www.csd.abdn.ac.uk/~apreece/Research/KRAFT.html>

Preece, A., Hui, K., & Gray, P. (1999). *KRAFT: Supporting virtual organisations through knowledge fusion. artificial intelligence for electronic commerce* (Tech. Rep. WS-99-01). AAAI Press.

Seligman, L., & Rosenthal, A. (1996). A metadata resource to promote data integration. Retrieved February 15, 2004, from <http://www.computer.org/conferences/meta96/seligman/seligman.html>

Sheth, A. P., & Larson, J. A. (1990). Federated database systems for managing distributed, heterogeneous and autonomous databases, *ACM Computing Surveys*, 22(3), 183-236.

Siegel, M. & Madnick, S. E. (1991). A metadata approach to resolving semantic conflicts. *Proceedings of the 17th Conference on Very Large Data Bases*, Barcelona, Spain, September.

Tzitzikas, Y., Spyratos, N., & Constantopoulos, P. (2001). Mediators over ontology-based information sources. *Proceedings of the 2nd International Conference on Web Information Systems Engineering*, Kyoto, Japan.

Visser, P. R. S., Jones, D. M., Bench-Capon, T. J. M., & Shave, M. J. R. (1998). Assessing heterogeneity by classifying ontology mismatches. *Proceedings of the International Conference on Formal Ontology in Information System* (pp. 148-162).

Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., et al. (2001). Ontology-based integration of information: A survey of existing approaches. *Proceedings of the IJCAI-01 Workshop: Ontologies and Information Sharing*, Seattle, WA.

Woelk, D., Cannata, P., Huhns, M., Shen, W., & Tomlinson, C. (1993). Using carnot for enterprise information integration. *Proceedings of the 2nd International Conference of Parallel and Distributed Information Systems* (pp. 133-136).

Woelk, D., & Tomlinson, C. (1994). The InfoSleuth Project: Intelligent Search Management via Semantic Agents (Tech. Rep.). Retrieved February 15, 2004, from <http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/Searching/woelk/woelk.html>

Yang, H. Cui, Z., & O'Brien, P. (1999). Extracting ontologies from legacy systems for understanding and re-engineering. *IEEE International Conference on Computer Software and Applications*.

## KEY TERMS

**Data Integration:** Process of unifying data that share some common semantics but originate from unrelated sources.

**Distributed Information System:** A set of information systems physically distributed over multiple sites, which are connected with some kind of communication network.

**Federated Database:** Idem FIS, but the information systems only involve databases (i.e., structured sources).

**Federated Information System (FIS):** A set of autonomous, distributed and heterogeneous information systems, which are operated together to generate a useful answer to users.

**Heterogeneous Information System:** A set of information systems that differs in syntactical or logical aspects, such as hardware platforms, data models or semantics.

**Ontological Changeability:** The ability of changing some structures of an information source without producing substantial changes in the ontological components of the integrated system.

**Ontological Reusability:** The ability of creating ontologies that can be used in different contexts or systems.

**Ontological Scalability:** The ability of easily adding new information sources without generate substantial changes in the ontological components of the integrated system.

**Ontology:** Provides a vocabulary to represent and communicate knowledge about the domain and a set of relationship containing the terms of the vocabulary at a conceptual level.

**Semantic Heterogeneity:** Each information source has a specific vocabulary according to its understanding of the world. The different interpretations of the terms within each of these vocabularies cause the semantic heterogeneity.



# Open Source Database Management Systems

**Sulayman K. Sowe**

*Aristotle University of Thessaloniki, Greece*

**Ioannis Samoladas**

*Aristotle University of Thessaloniki, Greece*

**Ioannis Stamelos**

*Aristotle University of Thessaloniki, Greece*

## INTRODUCTION

This article discusses open source database management systems (OSDBMS) trends from two broad perspectives. First, the software engineering discipline platform on which databases are built has recently witnessed a new form of software development—Free/Open Source Software Development (F/OSSD). Methodically, the F/OSSD paradigm has changed the way relational databases, initiated in the 1960s and 1970s, are developed, distributed, supported, and maintained. Second, commercial relational database management systems (RDBMS) still dominate the database market because, on one hand, vendors and users are skeptical of the boon of applications developed and distributed under the F/OSSD paradigm, and on the other hand, it has been argued that OSDBMS are not likely to follow the successful trend of other robust Free/Open Source Software (F/OSS) systems (Linux, Apache, etc.). This article presents trends in OSDBMS by looking at the morphology and landscape of the type of applications developed by the F/OSS community. Implementation of F/OSS strategies and factors mitigating the adoption and utilization of OSDBMS are explored by looking at the interactions between the F/OSSD process and database firms, vendors, and users.

The greatest claim levied against the F/OSS movement is that application development is tilted towards products that are highly modular and application program interfaces (APIs) rather than end-user products such as databases. The other thought, mostly coming from commercial RDBMS users, vendors, and enterprises claims that databases are a critical component of software to be left to the F/OSSD model. Both camps seem to have ample evidence to protect their stance. What we present here is a compilation of tenet views coming from existing literature presented in research and narrative journals and our experience as F/OSS participants.

## BACKGROUND

Users of F/OSS, having access to the source code, are free to study what the program does, modify it to suit their needs, distribute copies to other people, and publish improved versions so that the whole F/OSS community can benefit. The license agreement under which the source code is distributed exactly defines the rights users have over the product. Methodically, F/OSS is collaboratively developed software. An egalitarian network of developers, referred to as hackers, develop software online in a decentralized environment free of hierarchical control structures. Participants rely on extensive peer collaboration through the Internet, using project's mailing lists, e-mails, discussion forums, and so forth. However, communities have been found to provide support services such as suggestions for product features, acting as distributing organs, answering queries, helping new members, and so forth. Products at various development statuses (mature, stable, alpha, etc.) can be freely downloaded, while nonprofit foundations (NPF) and second-generation companies (SGC) may distribute products at minimal cost. F/OSSD has emerged as an alternative approach to decrease cycle-time, design complexities, improve software quality, and costs for a large number of software applications such as OSDBMS.

What is more interesting about the F/OSS landscape is that applications developed by the community are not uniformly distributed across all disciplines. Part of our ongoing survey of F/OSSD activities from both freshmeat.net (Figure1) and sourceforge.net (Figure2) shows that F/OSS projects cover wide ranging topics, with development dominated by infrastructural or Internet-based components. In this study, databases show a moderate increase. The Freshmeat study shows 17.18% increase in database-related projects during 20 weeks of our monitoring in 2003. Sourceforge recorded 24.20% during the same period. Popularity with the developer community shows that OSDBMS are no longer



Figure 1. *freshmeat.net*

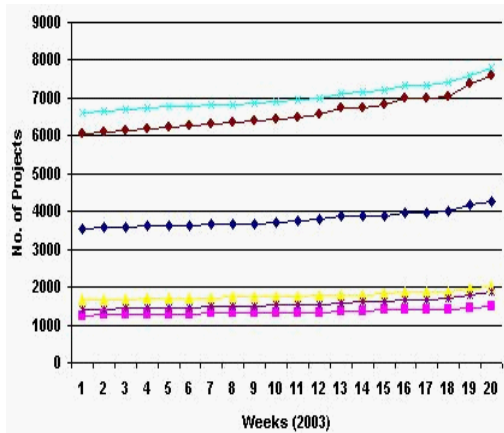
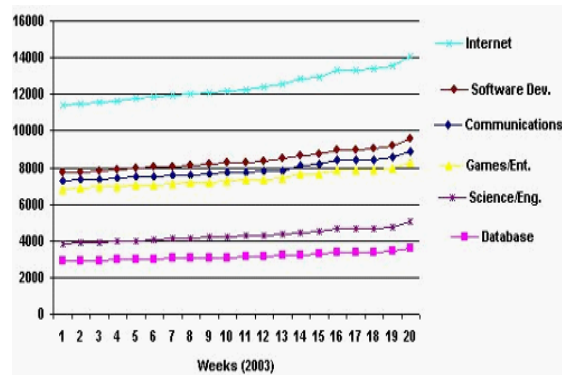


Figure 2. *sourceforge.net*



viewed as an add-on when distributing popular F/OSS products.

One explanation for the concentration of applications in the areas described could be that the F/OSSD model tends to work best for areas where developers are themselves users. The model has had much less success with regards to OSDBMS. In addition, F/OSSD has been viewed as a kind of intrinsic activity concentrating more on personally interesting aspects of software targeting the high-end technical user (Krishnamurthy, 2003). This leaves the development of most databases to commercial entities.

A database is a coherent and structured collection of related data, and OSDBMS represents those RDBMS built and distributed by means of the F/OSS model of software development and distribution. OSDBMS users not only have access to the product’s source code, which is not possible under commercial RDBMS, but may freely download and modify the code and tailor it to fit their particular needs. The F/OSSD model yields two kinds of OSDBMS. Pure OSDBMS emanates from the F/OSSD practices of a particular individual, research/academic institution, or a group of hackers with no corporate involvement. Hybrid OSDBMS, on the other hand, result from outsourcing a whole or part of a database development to the F/OSS community by a software firm.

OSDBMS are in use in all aspects of life and industry, ranging from manipulation of data for hospital use, accounting, scientific and engineering applications, to such areas as immigration, allowing data harmonization across geographical boundaries. The dawn of the knowledge economy has made them an important resource that can be harnessed to generate crucial information for critical and real-time decisions making. Table 1 shows a list of selected databases that are reported to be free of charge for academic or personal use (Jeusfeld, 2003). Heteroge-

neous communities of F/OSS developers develop these databases, targeting diverse aspects and needs.

## MAIN THRUST OF THE ARTICLE

The emergence of F/OSS enterprises seeks to push software development out of the academic stream into the commercial mainstream (Babcook, 2004; LaMonica, 2003b), and as a result, end-user applications such as OSDBMS are becoming more popular. Gedda (2003) substantiated that increased adoption of OSDBMS technologies by well-known organizations has also led to more enterprise recognition, hence, consideration of OSDBMS. What is more compelling is that companies (e.g., Sybase, Oracle, Sun, IBM) are increasingly implementing *open source strategies*—porting programs and application into the Linux environment while at the same time realizing that they can “charge complementary services” such as post-sale services (ibid). Thus, F/OSS is redefining the software industry (Feller & Fitzgerald, 2000) in general and database development in particular. There is a gradual shift in focus from protecting software knowledge to maximizing gain from F/OSSD, use, and distribution. Furthermore, software enterprises are realizing that there is the need to move from being in-house software developers and distributors to that of a service industry where software products are judged by their quality, reliability, and performance by the people who develop and use the software.

Beyond the promise of a reduced total cost of ownership of the software and potentially better support, there is an added dimension of freedom from *vendor lock-in*, the situation where an entire database application becomes dependent on a single vendor’s implementation of a technology (Adams, 2002). As conjectured by

## Open Source Database Management Systems

Table 1. Some databases available free of charge for academic or personal use

	Database	License Terms	Supplier / community	Description	OS support	Targeted Audience
1	<a href="#">Backplane Open Source Database</a>	<a href="#">Free (GPL/OSI compatible) and Commercial</a>	Backplane	A replicated database for Linux and FreeBSD which allow database operations over WAN latencies while maintaining full transactional coherency. Utility ( <a href="#">dcrmake</a> ) needed to build it	Linux, FreeBSD	Network
2	<a href="#">Galax</a>	<a href="#">LPC and Commercial</a>	AT & T and Bells labs, Lucent Tech	Implementation of the Xquery which supports XML Schema and static typing, and also comes with some optimisation features.	Win, Linux, Solaris, O'Caml compiler required	Students and researchers interested in XML Schema and query
3	<a href="#">Microsoft SQL Server Desktop Engine (MSDE)</a>		Microsoft	Ideal for client applications requiring an embedded database, new developers learning how to build data-driven applications, and websites serving up to 25 concurrent users	Windows	Embedded databases, SQL Server Admins, tools and patches.
4	<a href="#">ObjectDB for Java</a>	<a href="#">Free* and Commercial</a>	<a href="#">ObjectDB</a>	An Object Database Management System (ODBMS) designed to handle efficiently databases of various sizes, ranged from a few KBs to hundreds of GBs. Embedded and Server editions have database failure recovery features for <b>mission critical applications</b> .	OS Independent	Sun's Java Data Objects (JDO), Storage, and Web Applications
5	<a href="#">PicoSQL</a>	<a href="#">GPL</a>	<a href="#">Picosoft</a>	Open-source SQL database system originally derived from a commercial product.	Win, Linux	Multilingual
6	<a href="#">BKD</a>	<a href="#">Copyrighted, student/trial version available</a>	Open University	Bayesian Knowledge Discoverer, a program designed to extract Bayesian Belief Networks (BBNs) from (possibly incomplete) databases.	Win	Knowledge Management, Data Mining
7	<a href="#">ConceptBase</a>	<a href="#">Other</a>	<a href="#">ConceptBase</a>	A deductive and object-oriented multi-user database intended for conceptual modelling and coordination in design environments.	UNIX OS Solaris 2.4 or higher	Engineering and Educational
8	<a href="#">Emdros</a>	<a href="#">GPL</a>	<a href="#">Emdros Community</a>	A text analysing and annotating database engine for linguistics, publishing, and text processing (retrieval).	Win, Linux, SunOS/Solaris	Developers, Text analyses, translation, Business Applications
9	<a href="#">KAON</a>	<a href="#">LGPL</a>	Univ. of Karlsruhe	An ontology management infrastructure targeted for business applications		<a href="#">Ontologies (biological)</a>
10	<a href="#">LEAP</a>	<a href="#">GPL</a>	LEAP community	A RDBMS used as an educational tool around the world to help students, and assist researchers and teachers as they study and teach databases.	Win, Linux, Solaris, SunOS, etc	Educational and Research establishment

Jeusfeld (2003), when OSDBMS users have access to source code, they are not forced into a perpetual upgrade cycle. A panel discussing trends in open source databases sees this freedom as having a big impact on the ecosystem of the company distributing the OSDBMS (Editingwhiz, 2003).

Even though F/OSS applications have made a giant leap in the server sector (e.g., Apache) and operating system and network environment (e.g., Linux), OSDBMS have yet to make a substantial break into the database market dominated by Microsoft, Oracle, and IBM, to mention a few. Nevertheless, both Wayner (2001) and the 2004 Database Development Survey by Evans Data Corporation demonstrated that, while companies continue to use their commercial databases, there is a recognized phase-shift towards OSDBMS, especially when new applications and major upgrades are needed. This is partly due to the attractive pricing of major OSDBMS (Gedda, 2003; Lowe, 2002; Martin, 2003), viable developer and support community, and ability to be easily integrated with other F/OSS tools and systems. However, a growing number of companies adopt a cautionary approach towards OSDBMS full utilization. The appre-

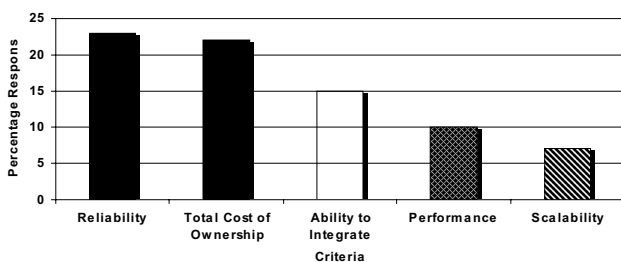
hensive trend is expected to continue, albeit OSDBMS offerings are continually improved. Overtly stated by Rooney (2004), the main challenges faced by large enterprises in adopting OSDBMS in mission-critical applications have constantly been scalability and third-party support. Despite success with the lower-end and mid-sized markets (Martin, 2003; Rooney, 2004), it may take a while for the low-cost attraction of OSDBMS to make a real impact in the database industry the way Linux has in the operating system. As argued by Krishnamurthy (2003), releasing the source code and pricing are two separate decisions. What is encouraging in this sense is that releasing the source code of a given OSDBMS only improves the innovation base of the system. Understandably, one can reinstall a crash operating system or restore a network server malfunction with little damage to a company's valuable data, but when a database application fails, a lot is at stake because databases contain information that is vital for the success of any industry in the information age. Yet, IT managers have reservations in trusting their companies most valuable data to OSDBMS (Wayner, 2001; Gedda, 2003).

## FUTURE TRENDS

Whereas commercial RDBMS vendors remain “locked in a ...fight for customers” (Boulton, 2003, p. 1), OSDBMS provide not only a less expensive and popular alternative to vendors and small to medium firms, but also become a dependable *Web content database* for many portals (Babcock, 2004; Brockmeier, 2003). The impetus has given rise to *xml* or *Web databases* that are tailored for the generation and storage of Web content. Many shared servers and low-budget Web sites (e.g., Yahoo.com, Slashdot.org, Sourceforge.net) used OSDBMS as an inexpensive option (Wayner, 2001). However, some Web sites had to find an alternative choice as their user base and volume of transaction increased. For example, Martin (2003) noted that the Sourceforge Web site had to migrate from PostgreSQL as a transaction database platform to DB2. OSDBMS have also been described as one of the pillars of today’s most powerful technologies, big and small businesses, and supporting high volume business transactions (LaMonica, 2003). Nonetheless, Figure 3 shows that for many executives, OSDBMS vendors, users, and developers, there are some urgent issues to consider when it comes to database selection. Other factors include readiness by many managing directors to accept the F/OSSD paradigm. One area where OSDBMS are making a breakthrough is in *grid computing*. With increase recognition of new research areas that are implementing OSDBMS in their work, OSDBMS mainstream adoption and use in mission-critical systems will be pervasive as early as 2006 (Babcock, 2004). According to Evans Data Corporation (2004), the reliability factor has persisted for some time as the most important criteria for selecting a database.

Embracing OSDBMS has remained a painful decision for database managers and administrators, and many users of commercial databases. A decision to select the right database depends on the needs of the organization and will be made based on budget, database size and

Figure 3. Most important criteria for selecting a database (Evans Data Corporation Database Development Survey, 2004)



scalability requirements, high-availability requirements, database functionality, and level of support required (Lowe, 2002). However, migrating to OSDBMS that cannot grow with customers’ needs could be costly for an enterprise in terms of retraining, infrastructure upgrades, licensing, and consulting fees. Rooney (2004) argued that these costs can be far greater than the initial savings of acquiring OSDBMS. Nonetheless, OSDBMS firms (e.g., sleepycat software) will continue to implement F/OSS strategies and make business by selling their commercial versions and offering support and other services (Boulton, 2003). At the same time, firms will embrace F/OSS paradigm in order to tap into “the vast global community of developers and reduce their cost of production by not having to [re]invent the wheel” (Krishnamurthy, 2003, p. 8). Furthermore, storage and security will define OSDBMS capability of the future, as data storage and computing needs of businesses continue to push database makers to develop faster and more scalable software (Orzech, 2003). Storage is an important factor because larger databases require larger storage facilities.

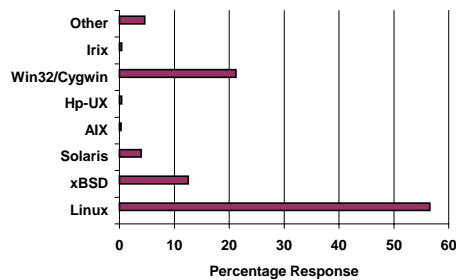
OSDBMS are no longer unknown quantities among the F/OSS community and software enterprises. Developers, vendors, and database users are already familiar with major players in the OSDBMS market—MySQL and PostgreSQL. Berkeley DB, Firebird, SAP DB, and many others also continue to attract attention in the database market, and the user base of these databases continues to grow.

What follows is a brief discussion on the two most popular OSDBMS. OSDBMS are ready for prime time because they:

- benefit from F/OSSD,
- are indeed scalable and popular with markets dominated by commercial databases,
- are suitable for industry’s mission-critical data (Gedda, 2003),
- reduce vendor lock-in, and
- as Boulton (2003) posits, are ripe for commoditization.

MySQL is considered as the most widely used OSDBMS that is constantly adding new features without comprising speed and reliability (Brockmeier, 2003). Available under General Public License (GPL) and commercial licences, MySQL is said to give database users and customers a choice. The dual licensing allows the use of MySQL as a commercial product without compromising its F/OSS status. The F/OSS community provides 24/7 services and support in terms of debugging, pool for patches, suggestions for future releases, and essential functionalities. The community gives users

Figure 4. Platforms on which PostgreSQL is used



and vendors the added confidence that resources are always available when problems need to be fixed and upgrades are needed, alleviating fear of vendor *lock-in*.

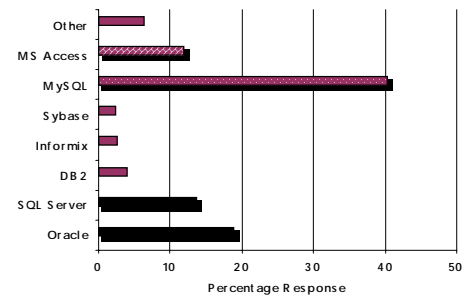
PostgreSQL is much earlier in the OSDBMS market than MySQL and has supported enterprise-level features (e.g., transaction support) much longer (Brockmeier, 2003). Released under Berkeley Software Distribution (BSD) licence, PostgreSQL like most OSDBMS can be freely downloaded from the project's Web site. Because its license allows companies to produce proprietary versions without requiring them to share their code with others or pay licensing fees, PostgreSQL is being perceived as more business friendly than its OSDBMS counterparts. However, Boulton (2003) posits that the license also allows companies to produce what may be incompatible versions of PostgreSQL. Even though this has not happen yet, there is a lot of forking going on in the F/OSSD scene.

The PostgreSQL user survey shows interesting characteristics of OSDBMS. As developers target the Linux operating system, a substantial OSDBMS run on Windows and other systems (Figure 4), reflecting the interests of the F/OSS developer and user communities as well as targeting market dominant operating systems. The survey also shows that there is a significant move towards OSDBMS from commercial RDBMS (Figure 5). Interestingly, there is also diffusion of users within OSDBMS. Over 40% of the 4,063 respondents used MySQL prior to using PostgreSQL.

## CONCLUSION

This paper has discussed and put forward thornier issues on the developments and impacts of OSDBMS on the operation of business organizations, educational and academic/research establishments, and socioeconomic and technological challenges posed by OSDBMS. Databases have evolved over time in development and design complexity, marketing strategies, and continue to have

Figure 5. Databases used prior to PostgreSQL



a greater impact in the way organizations operate and make their business decisions. The windfall has brought immense response from software companies, database researchers and professionals, database teachers and learners, and many more. While there is continued popularity in the use and implementation of OSDBMS, many continue to use their legacy databases. Thus, a form of *quasi modus vivendi* will continue to exist, at least in the immediate future, between OSDBMS and their commercial RDBMS counterparts, and the business and corporate culture needs to be aware of this symbiotic relationship. There are ample postulates about the success of F/OSS products and companies, but the success of OSDBMS on the same line needs a cautionary approach as the database market is not only small in scope but is getting increasingly saturated. The discussion presented here is our current understanding of the ecology of OSDBMS. A consolidated and ongoing empirical research is needed in OSDBMS, as has been done in other F/OSS projects, to better understand the trends, dual-license business model, sustainable F/OSS community involvement, and market factors surrounding the evolutionary trends of OSDBMS.

## REFERENCES

- Adams, L. (2002). Is vendor lock-in universally bad? Retrieved January 22, 2005, from <http://builder.com.com/5100-6387-1058909.html>
- Babcock, C. (2004). Popularity growing for open-source databases. Retrieved January 22, 2005, from <http://www.informationweek.com/shared/printableArticle.jhtml?articleID=18312009>
- Boulton, C. (2003). Are open source databases following in Linux footsteps? Retrieved January 22, 2005, from <http://www.databasejournal.com/news/article.php/2222061>



Brockmeier, J. (2003). Battle of the open source databases. Retrieved January 22, 2005, from <http://www.newsfactor.com/perl/story/20495.html>

Editingwhiz. (2003). Panel sees trends in open source for 2004. Retrieved January 22, 2005, from <http://software.itmanagersjournal.com/software/03/12/24/0155215.shtml>

Evans Data Corporation. (2004). Database development survey. Retrieved January 22, 2005, from [http://www.evansdata.com/n2/surveys/db\\_toc\\_04\\_1.shtml](http://www.evansdata.com/n2/surveys/db_toc_04_1.shtml)

Feller, J., & Fitzgerald, B. (2000). A framework analysis of the open source software development paradigm. In *Proceedings of the 21st International Conference on Information Systems (ICIS)*, Brisbane, Queensland, Australia (pp. 58-69).

Gedda, R. (2003). Analysis: Open source databases. Retrieved January 22, 2005, from <http://www.linuxworld.com.au/index.php?id=787941582&fp=2&fpid=1>

Jeusfeld, M. (2003). Publicly available database software. Retrieved January 22, 2005, from <http://www.acm.org/sigmod/databaseSoftware/>

Krishnamurthy, S. (2003). A managerial overview of open source software. *Business Horizons*, 46(5), 47-56.

LaMonica, M. (2003). Luxury models face cost-conscious buyers. Retrieved January 22, 2005, from [http://news.com.com/2009-1001\\_3-1001340.html](http://news.com.com/2009-1001_3-1001340.html)

Lowe, S. (2002). Database wars: Open source versus commercial. Retrieved January 22, 2005, from <http://www.zdnet.com.au/insight/0,39023731,20264039,00.htm>

Martin, V. (2003). Why DB2 vs Open source database sales guide. Retrieved January 22, 2005, from <ftp://ftp.software.ibm.com/software/data/pubs/papers/db2openspace.pdf>

Orzech, D. (2003). Rapidly falling storage cost means bigger databases, new applications. Retrieved January 22, 2005, from <http://www.cioupdate.com/trends/article.php/2217351>

Perez, J. (2004). Open source DBs go big time. Retrieved January 22, 2005, from [http://www.intelligententerprise.com/print\\_article.jhtml?articleID=17601165](http://www.intelligententerprise.com/print_article.jhtml?articleID=17601165)

Rooney, P. (2004). *Database, security, storage are next layers for open source commoditization*. Retrieved January 22, 2005, from <http://www.crn.com/Components/printArticle.asp?ArticleID=47322>

Sol, S. (2003). What is a database? (Part 1 of 4). Database articles and tutorials. Retrieved January 22, 2005, from <http://www.theukwebdesigncompany.com/articles/database.php>

Wayner, P. (2001). Open source databases bloom. Retrieved January 22, 2005, from <http://www.computerworld.com/softwaretopics/software/story/0,10801,63629,00.html>

## KEY TERMS

**Free/Open Source Software (F/OSS):** Software whose source code, under certain license agreements, is freely available for modification, distribution, and innovation.

**Grid Computing:** A distributed computing setting which has the tendency to allow users to communicate and share resources without much worry about its origin.

**Open Source Databases:** A class of relational databases developed and distributed by means of the F/OSS development model.

**Outsourcing:** The purchasing or contracting of tasks, goods, or services to an external source by a software firm.

**Second-Generation Companies (SGC):** F/OSS companies (e.g., MySQL, Sleepycat Software) employing dual-licensing-based business model that supports F/OSS philosophy and methodology in a profitable and sustainable software development environment.

**Total Cost of Ownership (TOC):** The total cost associated with acquiring software. This may include, but is not limited to, code downloading time, installation, maintenance, training, and so forth.

**Vendor Lock-In:** A situation where a software product is dependent on a single vendor's implementation of a technology.

**Web Services:** Technologies that make application-to-application communication on the World Wide Web possible.



# Open Source Software and Information Systems on the Web

**Antonio Cartelli**

*University of Cassino, Italy*

## INTRODUCTION

Databases and systems for their management are today more and more important for individual and corporate applications. Furthermore the spreading of the Internet has made possible the achievement of centralized information systems, accessible from everywhere on the Net, both for querying data and for managing them.

A special role is played in this process by open source software and especially by some packages which can be linked all together to build online information systems.

It is well known that open source tools are today widely used by developers and programmers and that they are a valid and reliable alternative to proprietary software, but often a mortal terror prevents common people from using them.

In what follows, after a short survey of historical events leading to the creation of the most famous open source packages and the Open Software Foundation, a description of the author's experiences is proposed, so showing how, in some special cases, an effort in learning new topics and developing adequate skills can produce relevant effects in solving very different problems and in researching.

## BACKGROUND

In the late '60s the spreading of computing systems led many scientists to hypothesize that the target of shared computing and the consequent access to computer resources by everyone could be hit with the use of great centralized systems, equipped with multi-program, time-sharing and multiple-access operating systems. The MULTICS (Multiplexed Information and Computer Service) case is perhaps the best example in this respect; it was a project carried out by MIT, Bell Labs, and General Electric, which meant creating a huge machine providing computing power to everyone (Corbato, Saltzer, & Clingen, 1972). Notwithstanding the MULTICS project was abandoned, it introduced many seminal ideas in computing literature, and some computer scientists at Bell Labs worked on it and made a one-user version of the system for a PDP-7 (DEC minicomputer); they called this new

system UNICS (Uniplexed Information and Computing Service) but suddenly renamed it UNIX (Bach, 1986). It is well known that Ritchie's development of the C language, the rewriting of the whole UNIX system in this new language, and its distribution for free to universities and research centers made the fortune of this operating system, which was implemented and installed on a great variety of computing systems in a few years.

The development of LSI (large-scale integration) circuits in late '70s led to personal computing, i.e., to computers not very different for their architecture from minicomputers but more and more cheaper. Everyone could now have a computer for his/her own use and, as history demonstrates, the dream of great centralized systems for shared computing was abandoned.

The case of operating systems, in the author's opinion, is emblematic for the impulse that personal computing gave to proprietary software. It is well known, in fact, that two main operating systems became popular on PCs. The first one, MS-DOS by Microsoft Inc., for the IBM PC and all machines equipped with the 8088 CPU, and the second one, UNIX by various distributors, for high-level personal computers equipped with the Motorola 68000 CPU family (Tanenbaum, 1987); none of the above systems was for free or was freely available. In other words, the introduction of PCs didn't help computer science knowledge, until then mostly shared among scholars and researchers, to exit from laboratories, and only the efforts of a few people led to the creation of operating systems freely available with their source code.

Once more, UNIX is the reference example. Its transformation into a commercial product with a license prohibiting the public study of its source code led many universities (adopting it for their operating systems courses when it was free) to abandon the system. Some scholars, on the contrary, decided to entirely rewrite the kernel of the system, letting it open, and one of the most famous examples in this regard was MINIX, developed by A. S. Tanenbaum (Tanenbaum, Van Staveren, Keizer, & Stevenson, 1983).

Autonomously from Tanenbaum, Richard Stallman and Linus Torvald developed prototypes of freely available operating systems and decided to join their efforts for creating GNU/Linux (today better known as Linux); it was

available on the Net in 1991 (Beck et al., 1996), but in a few years it evolved (and is still evolving) and became so steady and reliable as to be a serious and valid alternative to Microsoft Network Server's software.

Operating systems were not the only software freely available or developed to be freely accessible with their source code, but the success of Linux (as an operating system) is very important for the spreading of many other initiatives (which used that OS for their implementation).

Faster and more efficient ways for accessing software became available with the Internet, and the use of open and/or free software (under special licenses like BSD, GPL, etc.) was made easier. New individuals and communities of developers worked on other software projects and adopted the same strategy of making freely available their source code; consortia like FSF (Free Software Foundation) and OSF (Open Software Foundation) were then created for helping people in defining standards, protecting their rights, and continuing the hard work of developers.

Among the various projects one can find on the Net, the following ones, having a relevant part in what follows, will be analyzed in greater detail: the Apache Web server, the PHP scripting language, and the PostgreSQL RDBMS.

The Apache Web server was developed by a group of scientists who left the NCSA System Development Group and was made freely available on the Net (the Apache Web site <http://www.apache.org/> is a good starting point for the downloading of the server software). Once compiled and the program started, the HTTP daemon looks for requests coming from the Net and creates system processes to answer to them. One of the main features of this Web server is its modularity and the chance for a Web server administrator to integrate special modules, enhancing the server functionalities within it.

PHP is a scripting language (the reference Web site is <http://www.php.net/>), making easy for Webmasters the creation of Web interactive pages (i.e., FORMS letting data go back from client to Web server). Very valued features of this software are its modularity, its embedding features, and the interaction it guarantees with most widely used RDBMSs.

The PostgreSQL RDBMS is a software tool for the management of tables and queries in a relational database (it is available from the Web site <http://www.postgresql.org/>) by means of the well-known SQL (Structured Query Language), granting special users an easy access to data. The project for this software was firstly developed at Berkeley when the first examples of DBMSs (database management systems) were analyzed and discussed and the relational model was compared with the hierarchical and network models and adopted for it (Bracchi, Martella, & Pelagatti, 1987).

From the remarks reported until now, it can be easily deduced that the Internet and the above tools made easy and cheap the creation of information systems accessible by general and special users for querying and managing the databases hosted on Web servers (Cartelli, 2004a). In other words a Web server on the Internet (now mostly a PC) with all the above tools installed and running and the right Web pages for the storing/retrieving functions accessing a database becomes very similar to a mainframe with its virtual terminal services.

## **OPEN SOURCE SOFTWARE AND EDUCATION: TWO CASES**

Many sites on the Net today allow access to open source software, and some among them host whole projects, already developed or still evolving, with their communities of developers. Users, scholars, and programmers can easily download from these sites the software they need and simply use it or can take part in their developmental projects. An example of the above sites, SourceForge (<http://www.sourceforge.net/>) is reported here.

Nevertheless one can be induced to create new applications if the already existing ones seem not adequate for special situations or particular problems.

The experiences reported below are good examples, in the author's opinion, of the need for planning and creating special information systems by means of the open source software: The first one concerns the instruments to be used for paleographic research and teaching, and the second one concerns the carrying out of a special e-learning platform. Both of them are based on the use of the Linux operating system, the Web server Apache, the PHP language, and the PostgreSQL RDBMS.

### **Open Source, DBMSs, and the Community of Paleographers**

The two information systems described here are the result of the author's cooperation with M. Palma, a professor of Latin paleography at the University of Cassino (Italy). The first system is devoted to the management of the data concerning women who wrote manuscripts in the Middle Ages (women copyists); the second one manages the bibliography of the manuscripts written in Beneventan, i.e., an ancient medieval script used in South Italy.

The main aim of the dynamic Web site (<http://edu.let.unicas.it/womediev/>) named *Women and Written Culture in the Middle Ages* (Cartelli, Miglio, & Palma, 2001) was to systematize the data emerging from the research on women copyists while leading to an instru-

ment helping scholars and students to find new elements for further studies.

The data appearing relevant to the scientific community were—for women: the name, the qualification (i.e., if she was a nun or a lay), and the date or the period she belonged to; and for manuscripts: the shelfmark (i.e., town, library, and number of the manuscript), the place where it was written, the date or the period it belonged to, the authors and titles of the texts, and the bibliography (or the source of information about the manuscript). Furthermore it appeared important to show for each woman the manuscript/s she wrote and vice versa and, if possible and available, at least an image of the copyist's hand.

One of the main features of the system is to have two separated sections: the first one being operated only by the editors (by means of special FORMS) so that they can insert, modify, and delete the stored data and ensure the scientific validity of the information reported; and the second one being at everyone's disposal to obtain the list of all women and manuscripts in the database or to make queries concerning women and manuscripts with specific qualifications.

The other information system the author made up, called BMB online (*Bibliografia dei Manoscritti Beneventani online*; <http://edu.let.unicas.it/bmb/>), emerged from the analysis of the following elements: (1) the data to be stored in the database, (2) the users who had to access the database and the operations they were allowed, (3) the query system, and (4) the dataflow. In what follows, the above elements are analyzed in a greater detail (Cartelli & Palma, 2004).

- a. The database structure lies on six tables: The first one is used for the data of contributors and scientific administrator/s; the second one contains the data of the materials to be analyzed and identifies the contributor who has to write the corresponding bibliographic cards; the third one is used to store the data of Beneventan manuscripts; the fourth table hosts the first part of the bibliographic data (i.e., the location, the author/s, the title, and every data concerning a publication quoting one or more manuscripts); in the fifth table the second part of the bibliographic data is stored (i.e., the manuscripts' ID codes and the abstracts with the reason for the quotation); and the sixth table is an electronic blackboard and makes easier the communication among the people involved in the collection of the bibliographical materials.
- b. The users accessing the database have different rights and powers: (a) the ones with the least rights are those who can only query the system; (b) at the next level are the contributors who are allowed bibliographic operations; (c) the scientific administrator/s follow, who can manage all the data in the

database and write, modify, and certify the bibliographic cards (also if this last operation can be done only once); and (d) at the top of the access pyramid is the system administrator, who can do all the operations allowed to the scientific administrator/s and can access the verified cards to modify or to delete them.

- c. Once the bibliographic cards are compiled by the contributors and verified by the scientific administrator/s, they can be queried by generic users, who can access them in four different ways: (a) by author, (b) by manuscript, (c) by contributor, and (d) by one or more words, or part of them, concerning the location, the author, the series, etc. of a given publication.
- d. When the system starts the first time, the database is empty and the system administrator has to input the data for at least a scientific administrator. Scientific administrator/s can then input the data for one or more contributors, so letting them access the system; he/she can also input the bibliographic material to be chosen/assigned to the contributors and can input by him/herself the bibliographic cards. When the contributor/s access the materials to work on, they can compile the bibliographic cards. At last the cards are analyzed and revised by the administrator/s so that they can be read by general users.

It has to be noted that the above information systems were used both for paleographic research and teaching, and there is common agreement among scholars and professors on the effects the above instruments had on studying manuscripts and on teaching. The analysis of students attending the paleography course agree, in fact, with the results of the studies on ICT supporting communities of learners: Students involved in the above experiences not only developed computing skills greater than the ones they could obtain in traditional computing literacy courses but also were immersed in a meta-cognitive environment, were submitted to cognitive apprenticeship strategies, and were involved in the discussion and evaluation of the procedures they took part in (in other words, they experienced all the elements of meaningful learning; Varisco, 2002).

### Misconceptions, Mental Schemes, and E-Learning

The project reported here started from the analysis of the problems the students meet while approaching scientific topics and is still in progress; the conclusions reported below are then only partial, as regards the students'



learning problems and the hypothesized solutions, but are supporting the author's idea of information systems use in research and education and consequently the adoption of open source software for their development.

It is well known that students often manifest wrong ideas which can be interpreted in at least two ways: (1) mental schemes, if only the coherence of the students' ideas in the interpretation of phenomena is considered (with no reference to scientific paradigms), and (2) pre-conceptions or misconceptions (when the students' ideas are compared and evaluated with respect to the right scientific paradigms; Driver & Erickson, 1983).

Studies carried out all over the world with differently aged people (from students to workers, professionals, and teachers) showed that (Cartelli, 2002):

1. A map of the disciplinary fields with the students' wrong ideas can be drawn.
2. A lot of strategies and instruments have been proposed until now to help students in overcoming their problems, and a good percentage of success has been measured when they were adopted (nevertheless, there is no systematic study on them).
3. Wrong ideas can persist in students' minds also after the adoption of the above instruments and strategies.

The author's experience in computer science (CS) basic courses led him to hypothesize that a special e-learning platform continuously monitoring the didactic process could make easier for the students the learning of the various topics, while giving to professors a powerful instrument for managing their teaching.

The information system the author planned and carried out (Cartelli, 2003) was very similar in its features to an e-learning platform. It offered together with a well-structured knowledge tree of the topics to be taught/learned and special auto-evaluation tests, integrated within the course pages, the following functions: (1) various communication areas implementing virtual environments for teachers/professors, tutors, and students, (2) a careful management of the students' evaluation and assessment tests, and (3) two functions for the analysis of: (a) the students' access to course materials and (b) the use they made of communication services.

The management of all information in the site was guaranteed from five user levels or selected accesses: the system administrator, professors, tutors, students, and, at last, didactic researchers and scholars (who could only retrieve the information on the students' access to the course materials).

The two information retrieval functions that were used for students' monitoring had the following features:

1. The first one reported the number of the accesses at the site's pages that a single student or group of students made till the query date (the numerical data were reported into the tree structure of the site).
2. The second one gave the sequence of the students' accesses at the Web site, ordered by date and hour of access. It could also report the messages the student left in the electronic blackboard, chat, forum, and case study areas and let the teachers compare all the data stored in the same time interval.

The system was experimented with by two different sets of students and had positive effects as regards student performance. There was, in fact, only a 20% of students' loss at the ending examinations, and more than 65% of them had positive if not excellent scores. But a careful analysis of the data stored in the database showed the limit of the system: The amount of data generated by the second set of students (350 subjects) made impossible for the professor the continuous monitoring of the didactic process.

On another hand, the analysis of the students' answers at the assessment tests (at the end of the courses) still showed the presence of misconceptions and wrong ideas in a relevant number of persons.

## **FUTURE TRENDS**

It is undoubted that in the future the tools the author adopted for carrying out the information systems and the information systems themselves will be improved in their features so that it is not easy to foresee future trends of the research or further systems which will be needed.

Nonetheless a survey of two research studies still in progress will be given in what follows.

As regards paleography, the main field of interest is the influence of the information systems developed for studying and teaching in communities of practice (CoPS), communities of learners (CoLs), and virtual communities. An instrument still evolving will be deeply analyzed: the open catalogue of manuscripts (i.e., an information system supporting ancient libraries in making available on the Net their manuscripts and catalogues; it has been adopted until now only from the Malatestiana Library in Cesena (FO), Italy).

As regards information systems helping students in overcoming their difficulties, the results of the author's experiences induced a plan to implement descriptive and inferential statistical functions in the system (i.e., data mining functions; Cartelli, 2004b). The reasons for this choice have to be explored in the following elements to be continuously monitored:



1. the change in the time of the features of a single student (by means of special indices describing his/her behaviors and learning styles),
2. the change in the time of the features of the students' groups, i.e., the features of the classes they belong to,
3. the change in the space of the features of students' groups (how different environments can influence the evolution of students' learning models).

In other words, in the author's opinion, data mining strategies (and the database management systems made available from open source software) can be applied to the analysis of the teaching-learning process to improve teaching management and students' results.

## CONCLUSION

The above experiences, as already said above, were possible because of the availability of the open source tools Linux, PHP, Apache, and PostgreSQL, but they were also due to the help the author had from the developers' community on more than one occasion (i.e., to overcome some difficulties he met while implementing the projects). As a consequence the success of open source software has to be explored not only in the reliability and steadiness of that software but also in the circulation and sharing of ideas and expertise the developers' community affords.

Furthermore, in the case of BMB online, it has to be noted that the online information system comes after an experience the faculty started in 1992 with BIBMAN, a MS-DOS program which reached its physical limits in 2001. The choice of open source software for its carrying out is mostly due to the problems the staff met in recovering the data stored until that data, because the proprietary structure of the program didn't allow the autonomous elaboration of those data.

The experiences described here are only a part of the work the author completed during the last decade with open source software and are producing great changes in both the ways of carrying out research and teaching; it is then in the author's opinion that the systems for data management developed until now will produce their best effects if they will be made freely available when at a stable and definite development stage.

## REFERENCES

Bach, M. J. (1986). *The design of the UNIX operating system*. Englewood Cliffs, NJ: Prentice Hall.

Beck, M., Böhm, H., Dziadzka, M., Kunitz, U., Magnus, R., & Verworner, D. (1996). *LINUX kernel internals*. Harlow, UK: Addison-Wesley.

Bracchi, G., Martella G., & Pelagatti, G. (1987). *Sistemi per la gestione di base dei dati*. Turin, Italy: Petrini.

Cartelli, A. (2002). Web technologies and sciences epistemologies. In E. Cohen & E. Boyds (Eds.), *Proceedings of IS + IT Education 2002 International Conference*, Santa Rosa, CA (pp. 225-238). Retrieved August 16, 2004, from <http://ecommerce.lebow.drexel.edu/eli/2002Proceedings/papers/Carre203Webte.pdf>

Cartelli, A. (2003). Misinforming, misunderstanding, misconceptions: What informing science can do. In E. Cohen & E. Boyds (Eds.), *Proceedings of IS + IT Education 2003 International Conference* (pp. 1259-1273). Santa Rosa, CA: Informing Science Institute.

Cartelli, A. (2004a). Open source software and information management: The case of BMB on line. In M. Khosrow-Pour (Ed.), *Proceedings of IRMA 2004 International Conference: Innovations Through Information Technology* (pp. 1023-1024). Hershey, PA: Idea Group.

Cartelli, A. (2004b). Action-guidance: An action research project for the application of informing science in educational and vocational guidance. *Issues in Informing Science and Information Technology*, 1(1), 763-772.

Cartelli, A., Miglio, L., & Palma, M. (2001). New technologies and new paradigms in historical research. *Informing Science*, 4(2), 61-66.

Cartelli, A., & Palma, M. (2004). BMB on line: An information system for paleographic and didactic research. In M. Khosrow-Pour (Ed.), *Proceedings of IRMA 2004 International Conference: Innovations Through Information Technology* (pp. 45-47). Hershey, PA: Idea Group.

Corbato, F. J., Saltzer, J. H., & Clingen, C. T. (1972). MULTICS—The first seven years. *Proceedings of AFIPS Spring Joint Computer Conference*, 36 (pp. 571-583).

Driver, R., & Erickson, G. (1983). Theories in action: Some theoretical and empirical issues in the study of students' conceptual frameworks in science. *Studies in Science Education*, 10, 37.

Tanenbaum, A. S. (1987). *Operating systems, design and implementation*. Englewood Cliffs, NJ: Prentice Hall.

Tanenbaum, A. S., Van Staveren, H., Keizer, E. G., & Stevenson, J. W. (1983). A practical tool kit for making portable compilers. *Communications of the ACM*, 26(9), 654-660.



Varisco, B. M. (2002). *Costruttivismo socio-culturale. Genesi filosofiche, sviluppi psico-pedagogici, applicazioni didattiche*. Rome: Carocci.

## KEY TERMS

**Apache Software Foundation:** Provides support for the Apache community of open-source software projects. The Apache projects are characterized by a collaborative, consensus-based development process, an open and pragmatic software license, and a desire to create high quality software that leads the way in its field. Most famous projects among them are: http (the well-known Web server), XML (instruments for the development of Web pages based on XML—Extended Markup Language), and Jakarta (Java server).

**Data Mining:** Analysis of data in a database using tools which look for trends or anomalies without knowledge of the meaning of the data. Data mining was introduced by IBM, which holds some related patents. The application of these strategies can require a data warehouse (i.e., a system for storing, retrieving, and managing a large amount of data).

**E-Learning Platform:** Although there are today many types of e-learning platforms (free or not, open or not) very similar in their features to information systems, they can accomplish (all together or one by one) the following tasks: (1) to be a CMS (content management system), guaranteeing access to didactic materials for the students; (2) to be an LMS (learning management system), where the use of learning objects makes easier the learning of a given topic; (3) to be a CSCLS (computer-supported collaborative learning system), which makes easier the use of collaborative and situated teaching/learning strategies; and (4) to build a virtual community of students, tutors, and professors using KM (knowledge management) strategies.

**Information System:** The set of all human and mechanical resources needed for acquisition, storing, re-

trieving, and management of the vital data of the system. With human resources are usually intended both the individuals involved in the use of the system and the procedures they have to carry out. With mechanical resources have to be intended both the hardware and software instruments to be used for the management of data.

**Open Source Software:** It is the software one can have at his/her disposal together with the source code. Its main feature is to be submitted to licenses obliging those who want to distribute that software, parts of it, or changes to its structure, to do it together with the source code. A special consortium, called Open Group (Open Software Foundation), has been founded with the following mission: to drive the creation of boundaryless-information flow.

**PHP Project:** The project started with the collection of scripts to be executed as cgi-bin scripts from a Web server and soon became a real language. PHP is now a widely used general-purpose scripting language (very close to C for its syntax) that is especially suited for Web development. Very valued features of this software are: (1) modularity of the interpreting language, which can be built as a module for Web servers and especially for the Apache; (2) embedding features of the language, which can coexist with HTML code in Web pages; (3) creation on the fly of HTML pages (they can be produced by the server depending on conditions emerging from clients' answers or choices); and (4) interaction with widely used RDBMSs, including MySQL and PostgreSQL.

**PostgreSQL Project:** The project had its beginnings in 1986 inside the University of California at Berkeley as a research prototype and in the 16 years since then has moved to its now globally distributed development model, the PostgreSQL RDBMS (formerly known as Postgres, then as Postgres95), with central servers based in Canada. The PostgreSQL Global Development Group is a community of companies and people co-operating to drive the development of PostgreSQL.

# Optimization of Continual Queries

Sharifullah Khan

National University of Sciences and Technology, Pakistan

## INTRODUCTION

Recent advances in the technologies have made it possible to access information from *Internet-scale distributed data sources* (Florescu, Levy & Mendelzon, 1998). However, finding the right information at the right time is difficult. *Update monitoring* (Seligman et al., 2000) is a technology that gathers relevant information and forwards it to users in a timely way. *Continual queries* (CQs) (Chen et al., 2000; Khan & Mott, 2002a; Liu, Pu & Tang, 1999, 2000) provide a significant toolkit for update monitoring. They are persistent queries that are issued once and then are run at regular intervals or when data change until a termination condition is satisfied. They are then removed from the system. They relieve users from having to revisit Web sites or other data sources and reissue their queries frequently to obtain new information that match their queries. A CQ is a typical SQL query having additional triggering and termination conditions. CQs are of two types: change-based and time-based. An example of a CQ is “notify me in the next six months whenever the Microsoft stock price drops by more than 5% from today level.”

## CQs OPTIMIZATION

CQs are particularly useful for an environment like the Internet, comprises a large amount of frequently changing information (Chen et al., 2000; Khan & Mott, 2002a; Liu, Pu & Tang, 1999). A CQs system needs to be able to support a large number of queries from a large number of users. This poses many difficulties through the Internet’s widespread distribution and massive use by a large population of users (Chen et al., 2000; Khan & Mott, 2002b). For example, the queries will impact on the local operations of the data source and could overload the data source with duplicate computation of common tasks, each occurring in multiple CQs. Since each data source’s communications and data processing capacity must be divided among all the users as the number of users grows, data servers may become swamped. In addition, these queries could also overload networking with data traffic, much of which may be superfluous. One approach to controlling this problem is to group queries so that they share their computation on the assumption that many queries have

a similar structure. For example, consider relations  $r_1(a,b,c)$  and  $r_2(x,y,z)$ . The following queries have similar structure:

$$Q_2 = \sigma_{a>10 \wedge b<5 \wedge c=x} (r_1 \bowtie r_2)$$

$$Q_3 = \sigma_{a>15 \wedge b<5 \wedge c=x} (r_1 \bowtie r_2)$$

$$Q_4 = \sigma_{a<10 \wedge b>5 \wedge c=x} (r_1 \bowtie r_2)$$

Grouping queries optimizes the evaluation of the query by executing common operations in the group of queries just once (Chen et al., 2000; Khan & Mott, 2002b; Roy et al., 2000). Moreover, it also avoids unnecessary query invocation over autonomous data sources on the Internet and reduces data traffic over the networks.

Additionally, after the initial evaluation we want a CQ to return only changes in the data source it queries. The simple and straightforward way to perform this is the *complete evaluation* approach. This approach uploads the complete results of the CQ rather than just updated results, which increases data transmission over the networks. An important approach to the optimization of CQs is therefore to employ *differential evaluation*. It means that a CQ is evaluated on the changes that have been made in the base data since its previous evaluation of the CQ after the initial evaluation. This approach reduces the data transmission in the subsequent evaluations of the query (Liu et al., 1996). Clearly, this technique is best suited to conditions where the number of changes is small relative to a much larger quantity of source data (Liu et al., 1996).

## Existing Systems

Query grouping on the basis of common operations is not new (Roy et al., 2000). However, the grouping of CQs raises new issues: (1) a CQs system has to handle a large collection of CQs due to the scale of the Internet; (2) CQs are not all made available to the system at the same time; (3) a user’s requests are unpredictable and may change rapidly; and (4) CQs in a group can have different triggering times. So, CQs groups are dynamic; they are continually changing as old queries are deleted and new ones are added. The frequent insertion and deletion of CQs in groups can make those groups inefficient over time and hence reduce overall system performance (Chen et al., 2000; Khan & Mott, 2002b). In this case, one or more

groups may require *dynamic re-grouping* to maintain their effectiveness.

Traditional grouping techniques (Roy et al., 2000) are not suitable for CQs because they are designed to find an optimal plan for a small number of queries, and all of the queries of a group are made available to the grouping system at the time of planning (Chen et al., 2000; Khan & Mott, 2002b). Traditionally, groups of queries are static since all of the queries are entered in and removed from the group at the same time. In addition, it is expensive to extend the traditional grouping technique to handle CQs (Chen et al., 2000; Khan & Mott, 2002b).

A technique has been designed to group CQs incrementally on the basis of *expression signatures* (Chen et al., 2000; Chen, DeWitt & Naughton, 2002). This technique can scale to a large number of queries, and it treats time-based and change-based queries uniformly. In it, queries are grouped over the server and the *group query* (i.e., the group representative) is evaluated against the data source while the other queries of the group are deduced from the result of the group query. However, the technique does not handle *dynamic re-grouping* to maintain the effectiveness of the groups.

The previous paragraphs have noted that monitoring updates in the open environment of the Internet without grouping and differential evaluation can easily overload the data management systems and networks. It is therefore desirable to apply optimization techniques to CQs. However, the existing approaches to query grouping are not adequate for CQs grouping. CQs grouping needs an approach that is dynamic and scalable. In addition, the approach must provide dynamic re-grouping to maintain the group effectiveness.

### Proposed Approach to CQs Optimization

This section proposes the grouping strategy and a new execution model for grouped CQs. This model is necessary since existing execution models are inappropriate for grouped CQs.

Our grouping strategy is incremental (Khan & Mott, 2002b). It means that the existing groups are considered as a possible choice for a new input query to join, rather than re-grouping all the queries in the system when a new query is added. The new input query is merged into an existing group if the query matches to that group; otherwise, a new group consisting of just the new query is created. For example, the CQ system receives  $Q_2$  initially. There is no existing group in the system, so a new group  $G_1$  is generated. The specification of  $G_1$  is equal to the specification of  $Q_2$ . After a while,  $Q_3$  is received in the CQ system, so it is merged in the  $G_1$  because  $Q_3$  matches to  $G_1$

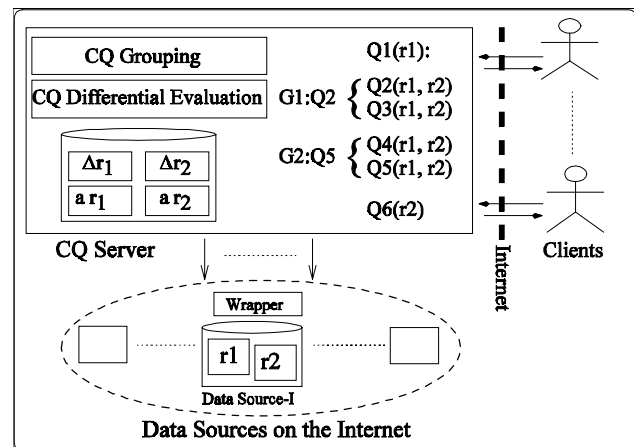
(i.e.,  $Q_3$  can be computed from  $G_1$ ). On other hand, at the arrival of  $Q_4$ , a new group  $G_2$  is generated because  $Q_4$  does not match to  $G_1$ .

In addition, there is then a dynamic step, whereby queries are added to and removed from the existing groups, thus causing the sizes of the groups to change, and all or some of the groups are re-grouped automatically. The technique is based on our new execution model of grouped CQs.

The new proposed execution model for grouped CQs is shown in Figure 1. It is based on differential evaluation of CQs. We maintain base data changes in differential relations such as  $\Delta r_1, \Delta r_2$ , which are held on the CQ server to make the system scalable (Liu et al., 1996). Each CQ is evaluated for the first time against the data sources. After the initial evaluation, single relation CQs are evaluated locally against these differential relations. However, multi-relations CQs can never be wholly evaluated against differential relations. They need to access data drawn from base relations which are known as auxiliary data and stored in auxiliary relations such as  $a r_1, a r_2$ , on the CQ server.

Multi-relation CQs are grouped on the CQ server. Each group of queries has its group query. A group query is just an ordinary query which is pruned of the projection operator to simplify the insertion of a new query into an existing group dynamically and to make grouping more scalable. To retrieve auxiliary data, we derive auxiliary query from a group query. These auxiliary queries are run when changes occur in differential relations (see more detail in Khan & Mott, 2002a). These data are then stored in auxiliary relations on the CQ server. Multi-relations CQs are evaluated against differential and auxiliary relations after an initial evaluation. Data are removed from the differential and auxiliary relations after the execution of all of the corresponding CQs.

Figure 1. New execution model for grouped CQs



### Discussion

A new execution model for grouped CQs has been proposed because dynamic re-grouping is expensive and wastes resources in the existing models. The fundamental features of this model are that group queries are pruned of projection operators and that the results of group queries are collectively stored in differential and auxiliary relations. Moreover, the model (1) avoids the replication of query results on the CQ server; (2) avoids proliferation of versions of group query results in dynamic re-grouping; (3) reduces storage consumption on the CQ server; and (4) makes our grouping technique scalable (Khan & Mott, 2002a) because the resultant data are stored collectively in differential and auxiliary relations on the CQ server and not according to a specific query/group schema.

Moreover, our grouping strategy is incremental and should therefore scale to a large number of queries. An advantage of the strategy is that it can be implemented using a general query engine without modification as an additional layer over typical grouping techniques. It also reduces data transmission in the subsequent executions of a CQ and improves query responsiveness (Khan & Mott, 2002b).

### REFERENCES

- Chen, J., DeWitt, D., et al. (2000). NIAGARACQ: A scalable continuous query system for Internet databases. *Proceedings of the ACM SIGMOD International Conference on the Management of Data* (pp. 379-390).
- Chen, J., DeWitt, D., & Naughton, J. (2002). Design and evaluation of alternative selection placement strategies in optimizing continuous queries. *Proceedings of International Conference on Data Engineering* (pp. 345-356).
- Florescu, D., Levy, A., & Mendelzon, A. (1998). Database techniques for the World Wide Web: A survey. *SIGMOD Record*, 27(3), 59-74.
- Khan, S., & Mott, P.L. (2002a). *LeedsCQ: A scalable continual queries system*. *Proceedings of 13th International Conference on Database and Expert Systems Applications*, Aix-en Provence, France.
- Khan, S., & Mott, P.L. (2002b). Scalable and dynamic grouping of continual queries. *Proceedings of the 2nd International Conference on Advances in Information Systems*, Izmir, Turkey.
- Liu, L., Pu, C., et al. (1996). Differential evaluation of continual queries. *Proceedings of the 16th International Conference on Distributed Computing Systems* (pp. 458-465).

Liu, L., Pu, C., & Tang, W. (1999). Continual queries for internet scale event-driven information delivery. *IEEE Transactions on Knowledge and Data Engineering*, 11(4), 610-628.

Liu, L., Pu, C., & Tang, W. (2000). WebCQ: Detecting and delivering information changes on the Web. *Proceedings of the 9th International Conference on Knowledge Management* (pp. 512-519).

Roy, P., Seshadri, S., et al. (2000). Efficient and extensible algorithms for multi query optimisation. *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 249-260).

Seligman, L., Lehner, P., Smith, K., et al. (2000). Decision-centric information monitoring. *Journal of Intelligent Information Systems*, 14, 29-50.

### KEY TERMS

**Change-Based CQs:** The CQs that are fired when new data arrives at a source.

**Complete Evaluation:** To re-evaluate a CQ on the whole base data (i.e., the new result) and then find the symmetric difference with the previous result set.

**Differential/Incremental Evaluation:** To re-evaluate a CQ on the changes that have been made in the base data since its previous evaluation.

**Group Query:** It is representative of a group, and the result of grouped queries is computed from the group query.

**Internet-Scale Distributed Data Sources:** Autonomous data sources connected through the Internet.

**Query Matching:** A query Q matches a group query G if and only if Q can be computed from the result of G, or G can be computed from the result of Q.

**Termination Condition:** The condition in a CQ which specifies to terminate the evaluation of the query.

**Time-Based CQs:** The CQs that are executed at regular intervals.

**Triggering Condition:** The condition in a CQ which specifies when to evaluate the query.



# Path-Oriented Queries and Tree Inclusion Problem

Yangjun Chen

University of Winnipeg, Canada

## INTRODUCTION

With the rapid advance of the Internet, management of structured documents such as XML documents has become more and more important (Marchiori, 1998). As a simplified version of SGML, XML is recommended by W3C (World Wide Web Consortium, 1998a; World Wide Web Consortium, 1998b) as a document description meta-language to exchange and manipulate data and documents on the WWW. It has been used to code various types of data in a wide range of application domains, including a Chemical Markup Language for exchanging data about molecules, the Open Financial Exchange for swapping financial data between banks and banks and customers, as well as a Geographical Markup Language for searching geographical information (Bosak, 1997; Zhang & Gruenwald, 2001). Also, a growing number of legacy systems are adapted to output data in the form of XML documents.

In recent years, efforts have been made to find an effective way to generate XML structures that are able to describe XML semantics in underlying relational databases (Chen & Huck, 2001; Florescu & Kossmann, 1999; Shanmugasundaram et al., 1999, 2000; Yoshikawa, Amagasa, Shimura, & Uemura, 2001). However, due to the substantial difference between the nested element structures of XML and the flat relational data, much redundancy is introduced; i.e., the XML data is either flattened into tuples containing many redundant elements or has many disconnected elements. Therefore, it is significant to explore a way to accommodate XML documents which is different from the relational theory. In addition, a variety of XML query languages have been proposed to provide a clue to manipulate XML documents (Abiteboul, Quass, McHugh, Widom, & Wiener, 1996; Chamberlin et al., 2001; Christophides, Cluet, & Simeon, 2000; Deutsch, Fernandez, Florescu, Levy, & Suci, 1988; Robie, Chamberlin, & Florescu, 2000; Robie, Lapp, & Schach, 1998). Although the languages differ according to expressiveness, underlying formalism, and data model, they share a common feature: *path-oriented queries*. Thus, finding efficient methods to do path matching is very important to evaluation of queries against huge volumes of XML documents.

## BACKGROUND

As a path-oriented language, XQL queries are represented by a line command which connects element types using path operators ('/' or '//'). '/' is the child operator which selects from immediate child nodes. '/' is the descendant operator which selects from arbitrary descendant nodes. In addition, symbol '@' precedes attribute names. By using these notations, all paths of tree representation can be expressed by element types, attributes, '/' and '@'. Exactly, a simple path can be described by the following Backus-Naur Form:

```
<simple path> ::= <PathOp> <SimplePathUnit> |
<PathOp><SimplePathUnit> '@' <AttName>
<PathOp> ::= '/' | '/'
<SimplePathUnit> ::= <ElementType> |
<ElementType><PathOp><SimplePathUnit>
```

The following is a simple path-oriented query:

```
/letter//body [para $contains$ 'visit'] (1)
```

where /letter//body is a path and [para \$contains\$ 'visit'] is a predicate, enquiring whether element "para" contains a word "visit."

Several paths can be jointed together using ^ to form a complex query as follows:

```
/hotel-room-reservation/name ?x ^
/hotel-room-reservation/location [city-or-district =
Winnipeg] ^
/hotel-room-reservation/location/address [street =
510 Portage Ave] (2)
```

## EVALUATION OF PATH-ORIENTED QUERIES

In this section, we show different ways to evaluate a path-oriented query. First, we discuss the basic methods used in a database environment. Then a new strategy for tree-inclusion, which can be embedded into a document



database to provide an efficient way to evaluate path-oriented queries, is discussed in great detail.

## QUERY EVALUATION BASED ON INVERSION

### Inversion on Elements and Words

There is a lot of work that considers using relational database techniques to store and retrieve XML documents, such as Arnold-Moore, Fuller, Lowe, Thom, and Wilkinson (1995); Florescu and Kossman (1999); and Zhang, Naughton, DeWitt, Luo, and Lohman (2001). Among them, the most representative is the method discussed in Zhang et al. In this method, two kinds of inverted indexes are established for text words and elements, by means of which a text word (or an element) is mapped to a list, which enumerates documents containing the word (or the element) and its position within each document. To speed up the query evaluation, the position of a word (or an element) is recorded as follows:

- $(Dno, Wposition, level)$  for a text word
- $(Dno, Eposition, level)$  for an element

where  $Dno$  is its document number,  $Wposition$  is its position in the document, and  $level$  is its nesting depth within the document;  $Eposition$  is a pair:  $\langle s, e \rangle$ , representing the positions of the start and end tags of an element, respectively. For instance, the document shown in Figure 1(a) is indexed as shown in Figure 1(b). The index for elements is called *E-index* and the index for words is called *T-index*.

Let  $(d, x, l)$  be an index entry for an element  $a$ . Let  $(d', x', l')$  be an index entry for a word  $b$ . Then,  $a$  contains  $b$  iff  $d = d'$  and  $x.s < x' < x.e$ . Let  $(d'', x'', l'')$  be an index entry for another element  $c$ . Then,  $a$  contains  $c$  iff  $d = d''$  and  $x.s < x''.s$  and  $x.e > x''.e$ . Using these properties, some simple path-oriented queries can be evaluated. For example, to process the query: `/hotel-room-reservation/location/[city-or-district = Winnipeg]`, the inverted lists of `hotel-room-reservation`, `location`, `city-or-district`, and `Winnipeg` will be retrieved and then their containment will be checked according to the above properties. In a relational database, E-index and T-index are mapped into the following two relations (note that primary keys are italicized):

- E-index (element, docno, begin, end, level)
- T-index (word, docno, wordPosition, level)

These index structures are efficient for simple cases, such as whether a word is contained in an element. However, in the case that a query is a nontrivial tree, the evaluation based on these index structures is an exponential time process. To see this, consider the query: `/hotel-room-reservation/location/address [street = Portage Ave.]`. To evaluate this query, four joins have to be performed. They are the self-joins on E-index relation to connect `hotel-room-reservation` and `location`, `location` and `address`, and `address` and `street`, as well as the join between E-index and T-index relations to connect `street` and `Portage Ave.` In general, for a document tree with  $n$  nodes and a query tree with  $m$  nodes, the checking of containment needs  $O(n^m)$  time using this method.

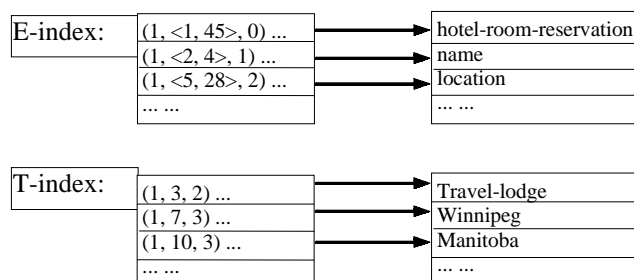
Figure 1. A sample XML file and its inverted lists

```

<hotel-room-reservation filecode="1302">
  <name>Travel-lodge</name>
  <location>
    <city-or-district>Winnipeg</city-or-district>
    <state>Manitoba</state>
    <country>Canada</country>
    <address>
      <number>500</number>
      <street>Portage Ave.</street>
      <post-code>R3B 2E9</post-code>
    </address>
  </location>
  <type>
    <room>one-bed-room</room>
    <price>$119.00</price>
  </type>
  <reservation-time>
    <from>April 28, 2003</from>
    <to>May 01, 2003</to>
  </reservation-time>
</hotel-room-reservation>

```

(a)



(b)

## Inversion on Paths and Words

The above method is improved by Seo, Lee, and Kim (2003) by introducing indexes on paths to reduce the number of joins as well as the sizes of relations involved in a join operation. This is achieved by establishing four relations to accommodate the inverted lists:

Path(path, pathID)  
 PathIndex(pathID, docno, begin, end)  
 Word(word, wordID)  
 WordIndex(wordID, docno, pathID, position)

In this way, the number of joins is dramatically decreased. For example, to process the same query: /hotel-room-reservation/location/address [street = Portage Ave.], only two joins are needed. The first join is between the Path and WordIndex relations with the join condition:

Path.path = '/hotel-room-reservation/location/address/street' ^  
 Path.pathID = WordIndex.pathID

The second join is between the result  $R$  of the first join and the Word relation with the join condition:

$R$ .wordID = Word.wordID ^  
 Word.Word = 'Portage Ave.'

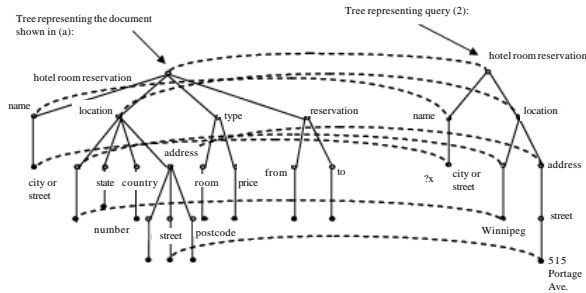
In general, the query evaluation based on such an index structure needs  $l$  joins, where  $l$  is the number of the words appearing in a query. However, such a time improvement is at the cost of memory space since in Path relation the element names are repeatedly stored. Concretely, for a document with  $n$  nodes, the size of the Path relation is on the order of  $O(n^2)$ . Therefore, the time complexity of this method is  $O(l \cdot d \cdot n^2)$ , where  $d$  represents the average length of paths.

## Query Evaluation Based on Tree Inclusion

As pointed out by Mannila and Raiha (1990), the evaluation of path-oriented queries is in essence a tree inclusion problem. For instance, to evaluate query (2), we will check whether there exists a document that contains the tree representing the query (see Figure 2 for illustration).

In the following, we first give a formal definition of tree inclusion. Then, a new algorithm for checking tree inclusion will be discussed.

Figure 2. Illustration for tree inclusion



**Definition 1 (tree inclusion):** Let  $P$  and  $T$  be rooted labeled trees. Let  $V(P)$  ( $V(T)$ ) be the set of the nodes in  $P$  ( $T$ ). We define an ordered embedding  $f: P \rightarrow T$  as an injective function  $f: V(P) \rightarrow V(T)$  such that for all nodes  $v, u \in V(P)$ ,

1.  $\text{label}(v) = \text{label}(f(v))$ ; (label preservation condition)
2.  $v$  is an ancestor of  $u$  iff  $f(v)$  is an ancestor of  $f(u)$ ; (ancestor condition)
3.  $v$  is to the left of  $u$  iff  $f(v)$  is to the left of  $f(u)$ ; (sibling condition)

For example, the tree representing the query (2) is included in the tree representing the document shown in Figure 1(a) (see Figure 2).

A lot of algorithms have been developed to check tree inclusion, such as Alonso and Schott (1993); Chen (1998); Kilpelainen and Mannila (1995); and Richter (1997). All these methods focus on the bottom-up strategies to get optimal computational complexities, but they are not suitable for database environment since the algorithms proposed assume that both the target tree (or, say, the document tree) and the pattern tree (or, say, the query tree) can be accommodated completely in the main memory. In the case of a large volume of data, it is not possible. Here we present a new algorithm by integrating a top-down process into a bottom-up computation, which has the same time complexity as the best bottom-up algorithm but needs no extra space. More importantly, it is more suitable for a database environment since by the top-down process each time only part of the tree is manipulated. Furthermore, it can be combined with *word signatures* to speed up query evaluation (Chen, 2003).

## Mixed Strategy for Tree Inclusion Problems

Now, we present our algorithm, which is designed based on the following three observations:

Algorithm A.

```

Function tree-inclusion( $T, P, a$ )                                (*top-down process*)
Input:  $T$  - a tree,  $S$  - a tree,  $a$  - an integer (at the very beginning,  $a$  is 0)
Output: ( $num, transnum$ ) - a pair of integers
begin
1. let  $r_1$  and  $r_2$  be the roots of  $T$  and  $P$ , respectively;
2. let  $T_1, \dots, T_k$  be the subtrees of  $r_1$ ;
3. let  $P_1, \dots, P_j$  be the subtrees of  $r_2$ ;
4. if  $label(r_1) = label(r_2)$  then
5.   {  $temp := \langle P_1, \dots, P_s \rangle; i := 1; j := 0; b := 0;$ 
6.   while ( $i \leq k \wedge temp \neq \emptyset$ ) do
7.     {  $x := forest-inclusion(T_i, temp, b);$ 
8.     if  $x.num > 0$  then {  $temp := temp / \langle P_{j+1}, \dots, P_{j+x.num} \rangle; j := j + x.num; b := 0;$ 
9.     else if ( $x.subnum =$  number of the subtrees of  $P_{j+1}$ 's root and  $label(T_i$ 's root) =  $label(P_{j+1}$ 's root))
10.      then {  $temp := temp / \langle P_{j+1} \rangle; j := j + 1; b := 0;$ 
11.      else  $b := x.transnum;$ 
12.       $i := i + 1;$  }
13.   if  $temp = \emptyset$  then return (1, 0);
14.   else { if  $j = 0$  then
15.     { if ( $b =$  number of the subtrees of  $P_1$ 's root and  $label(T$ 's root) =  $label(P_1$ 's root)) then  $j := 1;$ 
16.     if  $a = 0$  then return (0,  $j$ );
17.     else {  $x := forest-inclusion(T, \langle P_{a+1}, \dots, P_s \rangle, 0);$  return (0,  $\max\{j, a + x.num\}$ );} }
18.   }
19. else {  $b := 0;$ 
20.   for  $i = 1$  to  $k$  do
21.     {  $y := forest-inclusion(T_i, P, b); b := y.transnum;$ 
22.     if  $y.num = 1$  return (1, 0);}
23.   if  $a = 0$  then return (0,  $b$ );
24.   else {  $x := forest-inclusion(T, \langle P_{a+1}, \dots, P_s \rangle, 0);$  return (0,  $\max\{b, a + x.num\}$ )} }
25. }
end

```

1. Let  $r_1$  and  $r_2$  be the roots of  $T$  and  $P$ , respectively. If  $T$  includes  $P$  and  $label(r_1) = label(r_2)$ , we must have a root preserving embedding.
2. Let  $T_1, \dots, T_k$  be the subtrees of  $r_1$ . Let  $P_1, \dots, P_l$  be the subtrees of  $r_2$ . If  $T$  includes  $P$  and  $label(r_1) = label(r_2)$ , there must exist  $k_1, \dots, k_j$  and  $l_1, \dots, l_j$  ( $j \leq l$ ) such that  $T_{k_i}$  includes  $\langle P_{l_{i-1}+1}, \dots, P_{l_i} \rangle$  ( $i = 1, \dots, j; l_0 = 0$ ), where  $\langle P_{l_{i-1}+1}, \dots, P_{l_i} \rangle$  represents a forest containing subtrees  $P_{l_{i-1}+1}, \dots, P_{l_i}$ .
3. If  $T$  includes  $S$ , but  $label(r_1) \neq label(r_2)$ , there must exist an  $i$  such that  $T_i$  contains the whole  $S$ .

We notice that observation (1) and (3) hint a top-down process to find any possible root-preserving subtree embeddings while observation (2) hints a bottom-up process to find the left embedding for subforest inclusion. During the top-down process, the bottom-up process will be invoked to find the left embedding of  $\langle P_1, \dots, P_l \rangle$  in  $\langle T_1, \dots, T_k \rangle$ ; and during the bottom-up process, the top-down process may be invoked to find any possible root-preserving subtree embedding. (See Kilpelainen & Mannila, 1995, for the definitions of the root-preserving embedding and left emdedding.)

In a forest  $\langle T_1, \dots, T_k \rangle$ , the trees  $T_1, \dots, T_{i-1}$  are called the sibling trees of  $T_i$ .

Let  $P_{11}, \dots, P_{1q}$  be the subtrees of  $P_1$ 's root. The top-down process is designed as a function *tree-inclusion*( $T, P, a$ ), where  $0 \leq a \leq l$ . If  $a > 0$ , it indicates that the left sibling trees of  $T$  cover  $P_1, \dots, P_a$ . If  $T$  does not have any left sibling trees or the left sibling trees of  $T$  don't cover any of  $P_1, \dots, P_l$ ,  $a$  is set to 0. The output of *tree-inclusion*( $T, P, a$ ) is a pair of the form ( $num, transnum$ ), where  $num \in \{0, 1\}$  and  $a \leq transnum \leq q$ . If  $num = 1$ , it shows that  $T$  includes  $P$ . In this case, the value of *transnum* is not important and will be ignored. If  $num = 0$ , it shows that  $T$  does not include  $P$  but  $T$  covers  $P_a, \dots, P_{transnum}$ ; and thus  $T$ , together with its left sibling trees, covers  $P_1, \dots, P_{transnum}$ .

The bottom-up process is designed as a function *forest-inclusion*( $T, G, a$ ), where  $G = \langle P_1, \dots, P_s \rangle$  is a forest and  $0 \leq a \leq q$ . If  $a > 0$ , it indicates that the left sibling trees of  $T$  cover  $P_{11}, \dots, P_{1a}$ . If  $T$  does not have any left sibling trees or the left sibling trees of  $T$  don't cover any of  $P_{11}, \dots, P_{1q}$ ,  $a$  is equal to 0. The output of *forest-inclusion*( $T, G, a$ ) is a triplet of the form ( $num, subnum, transnum$ ), where  $0 \leq num \leq s$ ,  $0 \leq subnum \leq q$ , and  $a \leq transnum \leq q$ . If  $num > 0$ , *subnum*

and *transnum* are not important and can be ignored. If *num* = 0, *subnum* indicates that *T* covers  $P_{11}, \dots,$  and  $P_{1subnum}$  if *subnum* > 0; and *transnum* indicates that *T*, together with all its left sibling trees, covers  $P_{11}, \dots,$  and  $P_{1transnum}$ . We notice that the meaning of *subnum* and *transnum* are quite different. In the case that *num* = 0, *subnum* can be used to check whether *T* includes  $P_1$  by comparing *T*'s root and  $P_1$ 's root as well as *subnum* and the number of  $P_1$ 's children. If *T*'s root and  $P_1$ 's root have the same label and *subnum* and the number of  $P_1$ 's children are equal, we have *T* including  $P_1$ . In this way, a repeated checking of *T* against  $P_1$  can be avoided. On the other hand, *transnum* is used to avoid the checking of the tree rooted at *T*'s parent against  $P_1$  if it exists. Let *v* be the parent of *T*. Let *S* be the subtree rooted at *v* and  $S_1, \dots, S_k$  be the subtrees of *v* with  $T = S_i$  for some *i*. Assume that  $P_1$  is not included by any of  $S_1, \dots, S_k$ . Then, the return value of each  $g(S_i, G, a_i)$  ( $i = 1, \dots, k$ ) is of the form  $(0, subnum_i, transnum_i)$  and *transnum<sub>k</sub>* records the number of the subtrees in  $\langle P_{11}, \dots,$  and  $P_{1q} \rangle$ , which are covered by  $S_1, \dots, S_k$ . Thus, if  $label(v) = label(r_1)$  and *transnum<sub>k</sub>* = *q*, we know that *S* includes  $P_1$ .

First, we give the formal description of the top-down process (Algorithm A).

Algorithm A is made up of two parts. The first part contains lines 1-18. The second part contains lines 19-25

in Function *forest inclusion*(*S*, *G*, *b*). In the first part, we handle the case that  $label(r_1) = label(r_2)$ . In this case, we will perform a series of *forest-inclusion*( $T_i, G_i, a_i$ ) ( $i = 1, \dots, g$  for some  $g \leq k$ ), where  $G_i = \langle P_{l_i}, \dots, P_s \rangle$  with  $l_1 = 1 \leq l_2 \leq \dots \leq l_g \leq s$  and  $a_1 = 0 \leq a_2 \leq \dots \leq a_g \leq q$  (*q* is the number of the children of  $P_1$ 's root.) The return value of each *forest-inclusion*( $T_i, G_i, a_i$ ), is a triplet (*num*, *subnum*, *transnum*), according to which  $G_{i+1}$  and  $a_{i+1}$  are determined for a next call - *forest-inclusion*( $T_{i+1}, G_{i+1}, a_{i+1}$ ) as follows:

1. If *num* > 0, then  $G_{i+1} = \langle P_{l_i+num}, \dots, P_s \rangle, l_{i+1} = l_i + num,$  and  $a_{i+1} = 0$ . (See line 8.)
2. If *num* = 0, *subnum* = the number of the children of  $P_i$ 's root and  $label(T_i$ 's root) =  $label(P_i$ 's root), then  $G_{i+1} = \langle P_{l_i+1}, \dots, P_s \rangle, l_{i+1} = l_i + 1,$  and  $a_{i+1} = 0$ . (See lines 9-10.)
3. If *num* = 0, and *subnum* ≠ the number the children of  $P_i$ 's root or  $label(T_i$ 's root) ≠  $label(P_i$ 's root), then  $G_{i+1} = G_i, l_{i+1} = l_i,$  and  $a_{i+1} = transnum$ . (See line 11.)

When all  $T_i$ 's are checked, we will determine the return value of *tree-inclusion*(*T*, *P*, *a*). We distinguish between two cases:

Algorithm B.

```

Function forest-inclusion(S, G, b)                                (*bottom-up process*)
Input: S - a tree, G - a forest, b - an integer
Output: (num, subnum, transnum) - a triple of integers
begin
1. let  $r_1$  be the root of S; let  $S_1, \dots, S_k$  be the subtrees of  $r_1$ ;
2. let  $G = \langle P_1, \dots, P_s \rangle$ ;
3. find l such that  $|\langle P_1, \dots, P_l \rangle| \leq |S| < |\langle P_1, \dots, P_{l+1} \rangle|$ 
4. if  $l = 1$  and  $height(T_i) \geq height(P_1)$  then  $\{x := tree-inclusion(T, P_1, b);$  return  $(x.num, 0, x.transnum);\}$ 
5. if  $(l = 1$  and  $height(T_i) < height(P_1))$  or  $(l = 0)$  then
6. { let  $P_{11}, \dots, P_{1q}$  be the subtrees of  $P_1$ 's root;
7.    $x := forest-inclusion(S, \langle P_{1,b+1}, \dots, P_{1q} \rangle, 0);$ 
8.   return  $(0, 0, b + x.num);\}$ 
9. if  $l > 1$  then
10. {  $temp := \langle P_1, \dots, P_l \rangle;$     $i := 1; j := 0; c := 0;$ 
11.   while  $(i \leq k \wedge temp \neq \phi)$  do
12.   {  $x := forest-inclusion(S_i, temp, c);$ 
13.     if  $x.num > 0$  then  $\{temp := temp / \langle P_{j+1}, \dots, P_{j+x.num} \rangle;$   $j := j + x.num; c := 0;\}$ 
14.     else if  $(x.subnum = \text{number of the subtrees of } P_{j+1}'\text{s root and } label(T_i'\text{s root}) = label(P_{j+1}'\text{s root}))$ 
15.       then  $\{temp := temp / \langle P_{j+1} \rangle;$   $j := j + 1; c := 0;\}$ 
16.       else  $c := x.transnum;$ 
17.        $i := i + 1; \}$ 
18.   if  $(j > 0)$  then return  $(j, 0, 0)$ 
19.   else { if  $b = 0$  then return  $(0, c, c);$ 
20.     else  $\{x := forest-inclusion(S, \langle P_{j+1,b+1}, \dots, P_{j+1,l} \rangle, 0);$  return  $(0, c, \max\{c, b + x.num\});\}$ 
end

```

1. Let  $j$  be the number of subtrees in  $G$ , which are covered by  $\langle T_1, \dots, T_k \rangle$ . If  $j = s$ , return  $(num, transnum) = (1, 0)$ , indicating that  $T$  includes  $P$ . In this case,  $transnum$  is not important and is simply set to 0. (See line 13.)
2. If  $j < s$ , the return value is  $(0, transnum)$ , where  $transnum$  is determined as follows:
  - a. If  $j = 0$ , we will check whether  $b =$  number of the subtrees of  $P_1$ 's root and  $label(T$ 's root) =  $label(P_1$ 's root). If it is the case, set  $j$  to 1; otherwise,  $j$  remains 0. (See line 15.)
  - b. If  $j > 0$ , we will check whether  $a = 0$ . If it is the case, return  $(0, j)$ . Otherwise, we will call *forest-inclusion* $(T, \langle P_{a+1}, \dots, P_s \rangle, 0)$ . Assume that the return value is  $(num', subnum', transnum')$ . Then, the value of  $transnum$  is set to  $\max\{j, a + num'\}$ .

The second part handles the case that  $label(r_1) \neq label(r_2)$ . In this case, we will check each  $T_i$  ( $i = 1, \dots, k$ ) in turn to find any  $T_i$  which includes the whole  $P$ .

From the above analysis, we can see that if each time *forest-inclusion* $(T_i, G_i, a_i)$  returns a correct value, the result returned by *tree-inclusion* $(T, P, a)$  must be correct. Now we discuss *forest-inclusion* $(T_i, G_i, a_i)$  in detail. The following is its main idea:

1. Let  $G_i = \langle P_1, \dots, P_s \rangle$ . Find a  $j$  such that  $|T_i| \geq |\langle P_1, \dots, P_j \rangle|$  but  $|T_i| < |\langle P_1, \dots, P_{j+1} \rangle|$ .
2. If  $j > 1$ , we try to find an embedding of  $\langle P_1, \dots, P_j \rangle$  in the subtrees of  $T_i$ 's root. Let be  $T_{i1}, \dots,$  and  $T_{iz}$  be the subtrees of  $T_i$ 's root. Perform a series of *forest-inclusion* $(T_i, G_{ik}, b_k)$  ( $k = 1, \dots, c$  for some  $c \leq z$ ), where  $G_{ik} = \langle P_{l_k}, \dots, P_s \rangle$ , with  $l_1 = 1 \leq l_2 \leq \dots \leq l_c \leq j$  and  $a_1 = 0 \leq a_2 \leq \dots \leq a_g \leq q$  ( $q$  is the number of the children of  $P_1$ 's root).
3. If  $j = 1$ , i.e.,  $|T_i| \geq |P_1|$  but  $|T_i| < |\langle P_1, P_2 \rangle|$ , and  $height(T_i) \geq height(P_1)$ , call *tree-inclusion* $(T_i, P_1, a)$ . Assume that its return value is  $(num, transnum)$ . Then, the return value of *forest-inclusion* $(T_i, G_i, a_i)$  is set to be  $(num, 0, transnum)$ .
4. If  $j = 1$  but  $height(T_i) < height(P_1)$ , or  $j = 0$ , i.e.,  $|T_i| < |P_1|$ , call *forest-inclusion* $(T_i, \langle P_{1,a+1}, \dots, P_1q \rangle, 0)$ . Assume that its return value is  $(num, subnum, transnum)$ . Then, the return value of *forest-inclusion* $(T_i, G_i, a_i)$  is set to be  $(0, 0, transnum)$ .

From (2) and (4) shown above, we can see that this process is in essence a bottom-up process. However, it is not a pure bottom-up computation since if the condition in (3) is satisfied, a top-down process will be invoked.

Algorithm B is the formal description of the algorithm.

## FUTURE TREND

Document databases can be considered as well-organised information resources, which can be distributed over the Internet and become accessible to end users through the network. For this purpose, remote query evaluation has to be supported to replace the simple navigation along hyperlinks with the navigation through submitting specific queries. In this way, the search of information will be more efficient and more effective. However, the evaluation of remote queries is obviously more challenging than that of local ones, and much research on this is by all means required. Therefore, this must be one of the most important tasks in the near future.

## CONCLUSION

In this article, different methods for evaluating path-oriented queries are discussed. They are the inversion on elements and words (IEW), the inversion on paths and words (IPW), and the method based on a new tree-inclusion algorithm. In general, the IPW method has a better time complexity than the IEW, but it needs more memory space. In contrast, the tree inclusion needs no extra space but shows a better time complexity than both the IPW and the IEW. Especially, the signature technique can be integrated into the new tree-inclusion algorithm to cut off nonrelevant documents or nonrelevant elements as early as possible and improve the efficiency significantly.

## REFERENCES

- Abiteboul, S. Quass, D., McHugh, J., Widom, J., & Wiener, J. (1996). The Lorel query language for Ssemistructured data. *Journal of Digital Libraries*, 1(1).
- Alonso, L., & Schott, R. (1993). On the tree inclusion problem. In *Proceedings of Mathematical Foundations of Computer Science* (pp. 211-221).
- Arnold-Moore, T., Fuller, M., Lowe, B., Thom, J., & Wilkinson, R. (1995). The elf data model and ssql query language for structured document databases. In *Proceedings of the Australasian Database Conference*, Adelaide, Australia (pp. 17-26).
- Bosak, J. (1997). Java, and the future of the Web. Retrieved March 1997, from <http://sunsite.unc.edu/pub/sun-info/standards/xml/why/xmlapps.html>
- Chamberlin, D., Clark, J., Florescu, D., Robie, J., Simeon, J., & Stefanescu, M. (2001). Xquery 1.0: An xml query



- language. Technical report, *World Wide Web Consortium, 2001*. W3C Working Draft 07.
- Chen, M. (1998). More efficient algorithm for ordered tree inclusion. *Journal of Algorithms*, 26, 370-385.
- Chen, Y. (2003). Query evaluation and Web recognition in document databases. In *Proceedings IASTED International Conference on Internet and Multimedia Systems and Application (IMSA 2003)*, August 13-15, Honolulu (pp. 48-53).
- Chen, Y., & Huck, G. (2001). On the evaluation of path-oriented queries in document databases. *Lecture Notes in Computer Science* (Vol. 2113, 2001, pp. 953-962). Springer Verlag.
- Christophides, V., Cluet, S., & Simeon, J. (2000). On wrapping query languages and efficient XML integration. In *Proceedings of the ACM SIGMOD Conference on Management of Data* (pp. 141-152).
- Christodoulakis, S., & Faloutsos, C. (1984). Design consideration for a message file server. *IEEE Trans. Software Engineering*, 10(2), 201-210.
- Deutsch, A., Fernandez, M.F., Florescu, M.D., Levy, A., & Suciu, D. (1988). XML-QL: A Query Language for XML. Retrieved August 1988, from <http://www.w3.org/TR/NOTE-xml-ql>
- Faloutsos, C. (1985). Access methods for text. *ACM Computing Surveys*, 17(1), 49-74.
- Faloutsos, C. (1992). Signature files. In W.B. Frakes and R. Baeza-Yates (Eds.), *Information retrieval: Data structures & algorithms* (pp. 44-65). NJ: Prentice Hall.
- Florescu, D., & Kossman, D. (1999). Storing and querying XML data using an RDBMS. *IEEE Data Engineering Bulletin*, 22(3).
- Kilpelainen, P. & Mannila, H. (1995). Ordered and unordered tree inclusion. *SIAM Journal of Computing*, 24, 340-356.
- Mannila, H., & Raiha, K.-J. (n.d.). On query languages for the p-string data model. In H. Kangassalo, S. Ohsuga, & H. Jaakola (Eds.), *Information modelling and knowledge bases* (pp. 469-482). Amsterdam: IOS Press.
- Marchiori, M. (1998). The query languages workshop (QL'98), Retrieved from <http://www.w3.org/TandS/QL/QL98>
- Pixley, T. (2000). Document object model (DOM) level 2 events specification version 1.0. *W3C Recommendation*.
- Richter, T. (1997). A new algorithm for the ordered tree inclusion problem. In *Proceedings of the 8th Annual Symposium on Combinatorial Pattern Matching (CPM), in Lecture Notes of Computer Science (LNCS)* (Vol. 1264, pp. 150-166). Springer Verlag.
- Seo, C., Lee, S., & Kim, H. (2003). An efficient index technique for XML documents using RDBMS. *Information and Software Technology*, 45, 11-22. Elsevier Science B.V.
- Shanmugasundaram, J., Shekita, R., Carey, M.J., Lindsay, B.G., Pirahesh, H., & Reinwald, B. (2000). Efficiently publishing relational data as XML documents. In *Proceedings International Conference on Very Large Data Base (VLDB'00)* (pp. 65-76). Morgan Kaufmann Publishers.
- Shanmugasundaram, J., Tufte, K., Zhang, C., He, D.J., DeWitt, J., & Naughton, J.F. (1999). Relational databases for querying XML documents: Limitations and opportunities. In *Proceedings International Conference on Very Large Data Base (VLDB'99)* (pp. 302-314). Morgan Kaufmann Publishers.
- Suciu, D., & Vossen, G. (2000). *Proceedings of Third International Workshop on the Web and Databases (WebDB 2000)*, LNCS. Springer-Verlag.
- World Wide Web Consortium. (1998a). Extensible markup language (XML) 1.0. Retrieved from <http://www.w3.org/TR/1998/REC-xml/19980210>
- World Wide Web Consortium. (1998b). Extensible style language (XML). Working Draft, Dec. Retrieved from <http://www.w3.org/TR/1998/WD-xsl-19981216>
- Yoshikawa, M., Amagasa, T., Shimura, T., & Uemura, S. (2001) Xrel: A path-based approach to storage and retrieval of XML documents using relational databases. *ACM Transactions on Internet Technology*, 1(1).
- Zhang, C., Naughton, J., DeWitt, D., Luo, Q., & Lohman, G. (2001). On supporting containment queries in relational database management systems. In *Proceedings of ACM SIGMOD 2001*. ACM.
- Zhang, J., & Gruenwald, L. (2001). A GML-based open architecture for building a geographical information search engine over the Internet. In *Proceedings of WISE 2001* (pp. 385-392). IEEE.

## KEY TERMS

**Containment Query:** Queries that are based on the containment and proximity relationships among elements, attributes, and their contents.

**Document Database:** A database designed for managing and manipulating XML documents or even more generic SGML documents.

**Ordered and Labeled Tree:** Ordered labeled trees are trees whose nodes are labeled and in which the left-to-right order among siblings is significant.

**Path-Oriented Query:** Queries that are based on the path expressions including element tags, attributes, and key words.

**Signatures:** A signature is a hash-coded bit string assigned to key words used as indexes to speed up information retrieval.

**Tree Inclusion:** Given two ordered labeled trees  $T$  and  $S$ , the *tree inclusion problem* is to determine whether it is possible to obtain  $S$  from  $T$  by deleting nodes. Deleting a node  $v$  in tree  $T$  means making the children of  $v$  become the children of the parent of  $v$  and then removing  $v$ . If  $S$  can be obtained from  $T$  by deleting nodes, we say that  $T$  includes  $S$ .

**XML Document:** A document consisting of an (optional) XML declaration, followed by either an (optional) DTD or XML schema and then followed by document elements.

**XML Schema:** An alternative to DTDs. It is a schema language that assesses the validity of a well-formed element and attribute information items within an XML document. There are two major schema models: W3C XML Schema and Microsoft Schema.

# Preferred Repairs for Inconsistent Databases

**Sergio Greco**

*DEIS Università della Calabria, Italy*

**Cristina Sirangelo**

*DEIS Università della Calabria, Italy*

**Irina Trubitsyna**

*DEIS Università della Calabria, Italy*

**Ester Zumpano**

*DEIS Università della Calabria, Italy*

## INTRODUCTION

The objective of this article is to investigate the problems related to the extensional integration of information sources. In particular, we propose an approach for managing inconsistent databases, that is, databases violating integrity constraints. The problem of dealing with inconsistent information has recently assumed additional relevance as it plays a key role in all the areas in which duplicate information or conflicting information is likely to occur (Agarwal et al., 1995; Arenas, Bertossi & Chomicki, 1999; Bry, 1997; Dung, 1996; Lin & Mendelzon, 1999; Subrahmanian, 1994).

As known, the presence of inconsistent data can be resolved by repairing the database, that is, by providing a computational mechanism that ensures obtaining consistent “scenarios” of the information or by consistently answering queries posed on an inconsistent set of data.

### Example 1

Consider the following database schema consisting of the single relation *Teaches* (*Name, Faculty, Course*) with the functional dependency  $Name \rightarrow Faculty$ . Assume to have three different sources for the relation *Teaches* containing, respectively, the tuples *Teaches*(john, science, databases), *Teaches*(john, engineering, algorithms), and *Teaches*(john, science, operating\_systems). The three different source relations satisfy the functional dependencies, but from their integration, we derive the inconsistent relation  $D = \{(john, engineering, algorithms), (john, science, operating_systems), (john, science, databases)\}$ .

The integrated relation *D* can be repaired by applying a minimal set of update operations. In particular, it admits two repairs: *R1* obtained by deleting the tuple *(john, engineering, algorithms)* and *R2* obtained by deleting

the two tuples *(john, science, databases)* and *(john, science, operating\_systems)*.

In the presence of an alternative set of repairs, it is natural to allow user expressing preferences. In particular, methods for ranking and returning the preferred information is an increasingly key goal in AI applications ranging from information filtering and extraction to user profiling. The importance of this issue is reflected by an extensive number of proposals allowing the specification of preferences (Brewka & Eiter, 1999; Delgrande & Schaub, 2000; Gelfond & Son, 1997; Sakama & Inoue, 2000). This article is a contribution in this direction as it aims at filtering out the preferred repairs in the presence of preferences. For instance, if in the above example, preferred repairs are those minimizing the number of deletion and insertion of tuples, then the repair *R1* is preferred to the repair *R2*.

In this article, we consider preferences among repairs and possible answers by introducing a partial order among them on the base of some preference criteria. More specifically, preferences are expressed by considering polynomial functions applied to repairs and returning real numbers. The goodness of a repair is measured by estimating how much it violates the desiderata conditions, and a repair is *preferred* if it minimizes the value of the polynomial function used to express the preference criteria.

Note that, while integrity constraints can be considered as a query which must always be true after a modification of the database, the conditions expressed by the evaluation function can be considered as a set of desiderata which are satisfied *if possible* by a generic repair. The goodness of a repair is measured by estimating how much the updates to be performed on the inconsistent database respect the preference criteria or, in other words, how much the repaired database violates them.

The main contribution of this work consists in the proposal of a logic approach for querying and repairing inconsistent databases that extend previous works (Greco,

## Preferred Repairs for Inconsistent Databases

Greco & Zumpano, 2001; Greco & Zumpano, 2000) by allowing to express and manage preference criteria. The approach here proposed allows to express reliability on the information sources and is also suitable for expressing decision and optimization problems. The introduction of preference criteria strongly reduces the number of feasible repairs and answers; for special classes of constraints and functions, it gives a unique repair and answer.

## BACKGROUND

A database  $D$  has associated schema  $DS = (R_s, IC)$  which defines the intentional properties of  $D$ :  $R_s$  denotes the set of relation schemas whereas  $IC$  contains the set of integrity constraints. Integrity constraints express semantic information over data, that is, relationships that must hold among data in the theory. Generally, integrity constraints represent the interaction among data and define properties which are supposed to be explicitly satisfied by all instances over a given database schema. Therefore, they are mainly used to validate database transactions.

### Definition 1

A full (or universal) integrity constraint is a formula of the first order predicate calculus of the form:

$$(\forall X) [ B_1 \wedge \dots \wedge B_n \wedge \phi \supset A_1 \vee \dots \vee A_m \vee \psi_1 \vee \dots \vee \psi_k ]$$

where  $A_1, \dots, A_m, B_1, \dots, B_n$  are base positive literals,  $\phi, \psi_1, \dots, \psi_k$  are built-in literals,  $X$  denotes the list of all variables appearing in  $B_1, \dots, B_n$  and it is supposed that variables appearing in  $A_1, \dots, A_m, \phi, \psi_1, \dots, \psi_k$  also appear in  $B_1, \dots, B_n$ .

In the definition above, the conjunction  $B_1 \wedge \dots \wedge B_n \wedge \phi$  is called the *body* and the disjunction  $A_1 \vee \dots \vee A_m \vee \psi_1 \vee \dots \vee \psi_k$  the *head* of the integrity constraint. Moreover, an integrity constraint is said to be *positive* if no negated literals occur in it (classical definitions of integrity constraints only consider positive nondisjunctive constraints, called *embedded dependencies* (Kanellakis, 1991).

Often, we shall write our constraints in a different format by moving literals from the head to the body and vice versa.

## PREFERRED REPAIRS FOR INCONSISTENT DATABASES

In this section, we introduce a polynomial function, through which, expressing preferences criteria. The function introduces a partial order among repairs to allow the evaluation of the goodness of a repair for an inconsistent

database. Moreover, we define preferred repairs as feasible repairs that are minimal with respect to the partial order.

Let us first recall the formal definition of consistent database and repair.

### Definition 2

Given a database schema  $DS = (R_s, IC)$  we say that  $IC$  is consistent if there exists a database instance  $D$  over  $DS$  such that  $D \models IC$ . Moreover, we say that a database instance  $D$  over  $DS$  is consistent if  $D \models IC$ , that is, if all integrity constraints in  $IC$  are satisfied by  $D$ , otherwise it is inconsistent.

Intuitively, a *repair* for a (possibly inconsistent) database  $D$  is a minimal consistent set of insert and delete operations which makes  $D$  consistent, whereas a consistent answer for a query consists of two sets containing, respectively, the maximal set of true and undefined atoms which match the query goal; atoms which are neither true nor undefined can be assumed to be false.

More formally: see definition 3.

### Definition 3

Given a database schema  $DS = \langle R_s, IC \rangle$  and a database  $D$  over  $DS$  a repair for  $D$  is a pair of sets of atoms  $(R^+, R^-)$  such that 1)  $R^+ \cap R^- = \emptyset$ , 2)  $D \cup R^+ - R^- \models IC$  and 3) there is no pair  $(S^+, S^-) \neq (R^+, R^-)$  such that  $R^+ \subset S^+$ ,  $R^- \subset S^-$  and  $D \cup S^+ - S^- \models IC$ . The database  $D \cup R^+ - R^-$  will be called the *repaired database*.

Thus, repaired databases are consistent databases which are derived from the source database by means of a minimal (under total semantics) set of insertion and deletion of tuples. Given a repair  $R$  for  $D$ ,  $R^+$  denotes the set of tuples which will be added to the database, whereas  $R^-$  denotes the set of tuples of  $D$  which will be canceled. In the following, for a given repair  $R$  and a database  $D$ ,  $R(D) = D \cup R^+ - R^-$  denotes the application of  $R$  to  $D$ .

Moreover, given a database schema  $DS$ , we denote with  $D$  the set of all possible database instances over  $DS$ .

### Definition 4

Given a (possibly inconsistent) database  $D$  over a fixed schema  $DS$  and a polynomial function  $f: (D, D) \times D \rightarrow \mathfrak{R}$ . A repair  $R1$  is preferable to a repair  $R2$ , w.r.t. the function  $f$ , written  $R1 \ll_f R2$ , if  $f(R1, D) \leq f(R2, D)$ .

A repair  $R$  for  $D$  is said to be preferred w.r.t. the function  $f$  if there is no repair  $R'$  for  $D$  such that  $R' \ll_f R$ . A repaired

database  $D' = D \cup R^+ - R^-$  is said to be a preferred database if  $R = (R^+, R^-)$  is a preferred repair.

The above function  $f$  will be called (*repair*) *evaluation function* as it is used to evaluate a repair  $R$  with respect to a database  $D$ . A preferred database minimizes the value of the evaluation function  $f$  applied to the source database and repairs.

Observe that, in the above definition,  $D$  denotes the domain of all possible database instances whereas  $(D, D)$  denotes the domain of all possible database updates. This means that the evaluation function  $f$  can be used to measure any possible modification of the input databases and not only to measure repairs, that is, modification which makes the database consistent. In the following, for the sake of simplicity, we only consider functions which minimize the cardinality of a set.

## Example 2

Consider the database  $D = \{ \text{Teaches}(t1, c1), \text{Teaches}(t2, c2), \text{Faculty}(t1, \text{fac}1), \text{Faculty}(t2, \text{fac}1), \text{Course}(c1, \text{fac}1), \text{Course}(c2, \text{fac}2) \}$  and the integrity constraint

$$\forall (P, C, F) [ \text{Teaches}(P, C), \text{Faculty}(P, F) \supset \text{Course}(C, F) ]$$

Let  $R$  be a repair for the database  $D$ , possible evaluation functions are:

$$\begin{aligned} f1(R, D) &= |R^+| \text{ computing the number of inserted atoms,} \\ f2(R, D) &= |R^-| \text{ computing the number of deleted atoms,} \\ f3(R, D) &= |R^-| + |R^+| \text{ computing the number of deleted and inserted atoms.} \end{aligned}$$

As seen in Example 1, there are three repairs for  $D$ :

$$\begin{aligned} R1 &= (\emptyset, \{ \text{Teaches}(t2, c2) \}), \\ R2 &= (\emptyset, \{ \text{Faculty}(t2, \text{fac}1) \}), \text{ and} \\ R3 &= (\{ \text{Course}(c2, \text{fac}1) \}, \emptyset). \end{aligned}$$

With respect to the above evaluation functions, we have the following relations:

$$\begin{aligned} R1 &\ll_{f1} R3 \text{ and } R2 \ll_{f1} R3 \\ R3 &\ll_{f2} R1 \text{ and } R3 \ll_{f2} R2 \end{aligned}$$

Thus, considering the minimization of the above evaluation functions, we have that under the function  $f2$ ,  $R3$  is the unique preferred repair; under the function  $f1$ , we have two preferred repairs:  $R1$  and  $R2$ ; and under the function  $f3$ , all repairs are preferred.

Given a database  $D$ , a set of integrity constraints  $IC$  and an evaluation function  $f$ , we denote with  $\mathbf{R}(D, IC, f)$  the set

of preferred repairs for  $D$ . In the above example  $\mathbf{R}(D, IC, f1) = \{ R1, R2 \}$ , whereas  $\mathbf{R}(D, IC, f2) = \{ R3 \}$ . Moreover, we denote with  $f_0$  any constant evaluation function (e.g.,  $f_0(R, D) = 0$ , the function returning the value 0).  $\mathbf{R}(D, IC, f_0)$  denotes the set of all feasible repairs for  $D$  as no preference is introduced.

## Rewriting into Disjunctive Queries (Greco-Zumpano)

In Greco and Zumpano (2000), a general framework for computing repairs and consistent answers over inconsistent databases with universally quantified variables was proposed. The technique is based on the rewriting of constraints into extended disjunctive rules with two different forms of negation (negation as failure and classical negation). The disjunctive program can be used for two different purposes: compute repairs for the database and produce consistent answers, that is, a maximal set of atoms which do not violate the constraints. The technique is sound and complete (each stable model defines a repair, and each repair is derived from a stable model) and more general than techniques previously proposed.

More specifically, the technique is based on the generation of an extended disjunctive program  $LP$  derived from the set of integrity constraints. The repairs for the database can be generated from the stable models of  $LP$  whereas the computation of the consistent answers of a query  $(g, P)$  can be derived by considering the stable models of the program  $P \cup LP$  over the database  $D$ .

Let  $c$  be a universally quantified constraint of the form

$$\forall X [ B_1 \wedge \dots \wedge B_k \wedge \text{not } B_{k+1} \wedge \dots \wedge \text{not } B_n \wedge \phi \supset B_0 ]$$

then,  $dj(c)$  denotes the extended disjunctive rule

$$\begin{aligned} \neg B'_1 \vee \dots \vee \neg B'_k \wedge B'_{k+1} \vee \dots \vee B'_n \vee B'_0 \leftarrow (B_1 \vee B'_1), \dots, \\ (B_k \vee B'_k), (\text{not } B_{k+1} \vee \neg B'_{k+1}), \dots, (\text{not } B_n \vee \neg B'_n), \phi, \\ (\text{not } B_0 \vee \neg B'_0), \end{aligned}$$

where  $B'_i$  denotes the atom derived from  $B_i$ , by replacing the predicate symbol  $p$  with the new symbol  $p_d$  if  $B_i$  is a base atom; otherwise, it is equal to false. Let  $IC$  be a set of universally quantified integrity constraints, then  $DP(IC) = \{ dj(c) \mid c \in IC \}$  whereas  $LP(IC)$  is the set of standard disjunctive rules derived from  $DP(IC)$  by rewriting the body disjunctions.

Clearly, given a database  $D$  and a set of constraints  $IC$ ,  $LP(IC)_D$  denotes the program derived from the union of the rules  $LP(IC)$  with the facts in  $D$  whereas  $SM(LP(IC)_D)$  denotes the set of stable models of  $LP(IC)_D$ .



and every stable model is consistent since it cannot contain two atoms of the form  $A$  and  $\neg A$ . The following example shows how constraints are rewritten.

### Example 3

Consider the following integrity constraints:

$$\begin{aligned} \forall X [ p(X) \wedge \text{not } s(X) \supset q(X) ] \\ \forall X [ q(X) \supset r(X) ] \end{aligned}$$

and the database  $D$  containing the facts  $p(a)$ ,  $p(b)$ ,  $s(a)$ , and  $q(a)$ .

The derived generalized extended disjunctive program is defined as follows:

$$\begin{aligned} \neg p_d(X) \vee s_d(X) \vee q_d(X) \leftarrow (p(X) \vee p_d(X)) \wedge (\text{not } s(X) \\ \vee \neg s_d(X)) \wedge (\text{not } q(X) \vee \neg q_d(X)). \\ \neg q_d(X) \vee r_d(X) \leftarrow (q(X) \vee q_d(X)) \wedge (\text{not } r(X) \vee \neg r_d(X)). \end{aligned}$$

The above rules can now be rewritten in standard form. Let  $P$  be the corresponding extended disjunctive Datalog program. The computation of the program  $P_D$  gives the following stable models:

$$\begin{aligned} M_1 = D \cup \{ \neg p_d(b), \neg q_d(a) \}, M_2 = D \cup \{ \neg p_d(b), r_d(a) \}, \\ M_3 = D \cup \{ \neg q_d(a), s_d(b) \}, M_4 = D \cup \{ r_d(a), s_d(b) \}, \\ M_5 = D \cup \{ q_d(b), \neg q_d(a), r_d(b) \} \text{ and } M_6 = D \cup \{ q_d(b), \\ r_d(a), r_d(b) \}. \end{aligned}$$

A (generalized) extended disjunctive Datalog program can be simplified by eliminating from the body rules all literals whose predicate symbols are derived and do not appear in the head of any rule (these literals cannot be true). As mentioned before, the rewriting of constraints into disjunctive rules is useful for both (1) making the database consistent through the insertion and deletion of tuples and (2) computing consistent answers leaving the database inconsistent.

## Computing Database Repairs

Every stable model can be used to define a possible repair for the database by interpreting new derived atoms (denoted by the subscript “d”) as insertions and deletions of tuples. Thus, if a stable model  $M$  contains two atoms  $\emptyset \delta p_d(t)$  (derived atom) and  $p(t)$  (base atom), we deduce that the atom  $p(t)$  violates some constraints, and therefore, it must be deleted. Analogously, if  $M$  contains the derived atoms  $p_d(t)$  and does not contain  $p(t)$  (i.e.,  $p(t)$  is not in the database), we deduce that the atom  $p(t)$  should be inserted in the database. We now formalize the definition of repaired database.

Given a database schema  $DS = (Rs, IC)$  and a database  $D$  over  $DS$ , let  $M$  be a stable model of  $LP(IC)_D$ , then, a repair for  $D$  is a pair:

$$R(M) = ( \{ p(t) \mid p_d(t) \in M \wedge p(t) \notin D \}, \{ p(t) \mid \neg p_d(t) \in M \wedge p_d(t) \in D \} ).$$

Given a database schema  $DS = (Rs, IC)$  and a database  $D$  over  $DS$ , a repair for  $D$  is a pair of sets of atoms  $(R^+, R^-)$  such that 1)  $R^+ \cap R^- = \emptyset$ , 2)  $D \cup R^+ - R^- \models IC$ , and 3) there is no pair  $(S^+, S^-) \neq (R^+, R^-)$  such that  $S^+ \supset R^+$ ,  $S^- \supset R^-$  and  $D \cup S^+ - S^- \models IC$ . The database  $D \cup R^+ - R^-$  will be called the repaired database.

Thus, repaired databases are consistent databases which are derived from the source database by means of a minimal set of insertion and deletion of tuples. Given a repair  $R$  for  $D$ ,  $R^+$  denotes the set of tuples which will be added to the database whereas  $R^-$  denotes the set of tuples of  $D$  which will be canceled. In the following, for a given repair  $R$  and a database  $D$ ,  $R(D) = D \cup R^+ - R^-$  denotes the application of  $R$  to  $D$ .

The technique is sound and complete:

- (Soundness) for every stable model  $M$  of  $LP(IC)_D$ ,  $R(M)$ , there is a repair for  $D$ ;
- (Completeness) for every database repair  $S$  for  $D$ , there exists a stable model  $M$  for  $LP(IC)_D$  such that  $S = R(M)$ .

## FUTURE TRENDS

As a future trend, an interesting topic consists of investigating the specification of more flexible and powerful forms of preferences, that is, dynamic preferences that appear within the logic program and need to be determined “on the fly”.

## CONCLUSION

In this article, we have proposed a logic programming-based framework for managing possibly inconsistent databases. The main contribution of this work consists in the definition of a logic approach for repairing inconsistent databases that extends previous works by also considering techniques to express and manage preferences. The goodness of a repair is measured by estimating how much it violates the desiderata conditions and a repair is preferred if it minimizes the value of the polynomial function used to express the preference criteria. A further important characteristic related to the introduction of preference criteria is the reduction of feasible repairs and

answers, which lead, for special cases of constraints, to unique repair and answer.

## REFERENCES

- Abiteboul, S., Hull, R., & Vianu, V. (1995). *Foundations of databases*. Addison-Wesley.
- Agrawal, S., Keller, A.M., Wiederhold, G., & Sarawat, K. (1995). Flexible relation: An approach for integrating data from multiple, possibly inconsistent databases. *Proceedings of the IEEE International Conference on Data Engineering* (pp. 495-504).
- Arenas, M., Bertossi, L., & Chomicki, J. (1999). Consistent query answers in inconsistent databases. *Proceedings of the International Conference on Principles of Database Systems* (pp. 68-79).
- Brewka, G., & Eiter, T. (1999). Preferred answer sets for extended logic programs. *Artificial Intelligence*, 109(1-2), 297-356.
- Bry, F. (1997). Query answering in information system with integrity constraints. *Proceedings of the IFIP WG 11.5 Working Conference on Integrity and Control in Information Systems*.
- Delgrande, J.P., & Schaub, T. (2000). Expressing preferences in default logic. *Artificial Intelligence*, 123(1-2), 41-87.
- Dung, P.M. (1996). Integrating data from possibly inconsistent databases. *Proceedings of the International Conference on Cooperative Information Systems* (pp. 58-65).
- Eiter, T., Gottlob, G., & Mannila, H. (1997). Disjunctive datalog. *ACM Transactions on Database Systems*, 22(3), 364-418.
- Gelfond, M., & Lifschitz, V. (1991). Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 3(4), 365-386.
- Gelfond, M., & Son, T.C. (1997). Reasoning with prioritized defaults. *LPKR* (pp. 164-223).
- Grant, J., & Subrahmanian, V.S. (1995). Reasoning in inconsistent knowledge bases. *IEEE Transaction on Knowledge and Data Engineering*, 7(1), 177-189.
- Greco, G., Greco, S., & Zumpano, E. (2001). A logic programming approach to the integration, repairing and querying of inconsistent databases. *Proceedings of the International Conference on Logic Programming*.
- Greco, G., Sirangelo C., Trubitsyna I., & Zumpano, E. (2003). Preferred repairs for inconsistent databases. *Proceedings of the International Conference on Database Engineering and Applications Symposium*.
- Greco, S., & Zumpano, E. (2000). Querying inconsistent databases. *Proceedings of the International Conference on Logic Programming and Automated Reasoning* (pp. 308-325).
- Greco, S., & Saccà, D. (1990). Negative logic programs. *Proceedings of the North American Conference on Logic Programming* (pp. 480-497).
- Lin, J., & Mendelzon, A.O. (1999). Knowledge base merging by majority. In R. Pareschi & B. Fronhofer (Eds.), *Dynamic worlds: From the frame problem to knowledge management*. Kluwer.
- Minker, J. (1982). On indefinite data bases and the closed world assumption. *Proceedings of the 6th Conference on Automated Deduction* (pp. 292-308).
- Sakama, C., & Inoue, K. (2000). Prioritized logic programming and its application to commonsense reasoning. *Artificial Intelligence*, 123(1-2), 185-222.
- Subrahmanian, V.S. (1994). Amalgamating knowledge bases. *Proceedings of the ACM ToDS*, 19(2), 291-331.
- Ullman, J.K. (2000). Information integration using logical views. 239(2), 189-210.

## KEY TERMS

**Consistent Answer:** A set of tuples, derived from the database, satisfying all integrity constraints.

**Consistent Database:** A database satisfying a set of integrity constraints.

**Data Integration:** A process providing a uniform integrated access to multiple heterogeneous information sources.

**Disjunctive Datalog Program:** A set of rules of the form:

$$A_1 \vee \dots \vee A_k \leftarrow B_1, \dots, B_m, \text{not } B_{m+1}, \dots, \text{not } B_n, k+m+n > 0$$

where  $A_1, \dots, A_k, B_1, \dots, B_n$  are atoms of the form  $p(t_1, \dots, t_h)$ ,  $p$  is a predicate symbol of arity  $h$  and the terms  $t_1, \dots, t_h$  are constants or variables.

## *Preferred Repairs for Inconsistent Databases*

**Inconsistent Database:** A database violating some integrity constraints.

**Integrity Constraints:** Set of constraints which must be satisfied by database instances.

**Preferred Repair:** A repair minimizing the value of the evaluation function  $f$  applied to the source database.

**Repair:** Minimal set of insert and delete operations which make the database consistent.

P

# Proper Placement of Derived Classes in the Class Hierarchy

Reda Alhajj

University of Calgary, Canada

Faruk Polat

Middle East Technical University, Turkey

## INTRODUCTION

Users may derive new classes by defining views<sup>1</sup> based on the current database contents. Some virtual classes are classified as brothers of existing classes, and others are either superclasses or subclasses of existing base and virtual classes. A base class is defined directly by the user using class definition constructs. A virtual class is classified as a brother of another class if it is derived from the latter class via a selection. To have a homogeneous system, virtual classes must be treated as first-class citizens in an object-oriented model.

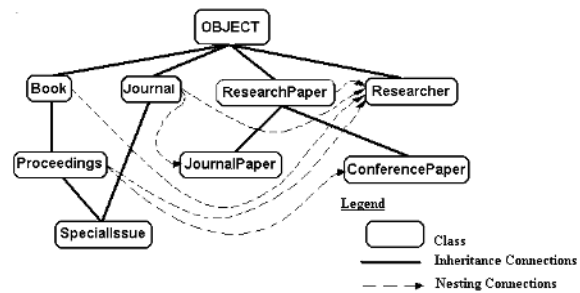
In this article, we handle reusability maximization by investigating the proper location of a derived class in the hierarchy by adjusting the list of superclasses and the set of subclasses for a given class to increase inheritance instead of duplication. Our aim is to maximize inherited, and minimize locally defined characteristics by adjusting the list of superclasses to include classes that maximize inherited and minimize locally defined characteristics.

Depending on the specification of their superclasses and subclasses during query evaluation, virtual classes may be classified into four groups: (1) classes for which both superclasses and subclasses are specified; (2) classes for which only superclasses are specified; (3) classes for which only subclasses are specified; and (4) classes for which neither superclasses nor subclasses are specified. Reusability decreases from the first group to the fourth group. Two algorithms are presented to achieve the target of maximizing reusability when possible.

## BACKGROUND

Shown in Figure 1 is a class hierarchy that will be referenced frequently in the article where illustrating examples are necessary. Given next is the basic terminology necessary to understand the rest of this article.

Figure 1. An example class hierarchy



1.  $P_d(c)$ : set of classes used in deriving class  $c$ ;  $P_d$  is empty for base classes because they are user defined.
2.  $C_p(c)$ : list of direct superclasses of class  $c$ .
3.  $C_b(c)$ : set of direct subclasses of class  $c$ .
4.  $L_{instances}(c)$ : set of object identifiers of objects added directly to class  $c$ .
5.  $W_{instances}(c)$ : set of object identities of objects in class  $c$  and its direct and indirect subclasses.
6.  $W_{attributes}(c)$ : set of attributes that determine the state of objects in  $L_{instances}(c)$ . It includes inherited and locally defined attributes.
7.  $L_{attributes}(c)$ : set of additional attributes locally defined in class  $c$ , that is,  $L_{attributes}(c) \subseteq W_{attributes}(c)$ .
8.  $W_{behavior}(c)$ : set of methods common to all objects in class  $c$ . It includes both inherited methods and locally defined methods.
9.  $L_{behavior}(c)$ : set of additional methods locally defined in class  $c$ , that is,  $L_{behavior}(c) \subseteq W_{behavior}(c)$ . For every virtual class  $c$ ,  $L_{behavior}(c)$  includes a method, called FindObjects(), which implements the query expression that decides on objects in  $W_{instances}(c)$ .
10.  $m(c, f, r)$ : the header of method  $m$  such that,  $m \in L_{behavior}(c)$ ,  $f$  is the list of domains for formal parameters, and  $r$  is the domain of the result.

## Proper Placement of Derived Classes in the Class Hierarchy

**Example 1:** [Base classes] All the classes shown in Figure 1 are base classes. Next are the characteristics of some of them, where B, J, P, SI, RP, R, JP, and CP stand for Book, Journal, Proceedings, SpecialIssue, ResearchPaper, Researcher, JournalPaper, and ConferencePaper, respectively:

$$C_p(B)=[\text{OBJECT}] \quad C_b(B)={P} \quad P_d(B)=\phi$$

$$L_{\text{attributes}}(B)={\text{title:String, author:[R], year:Date, subject:String, publisher:String}}$$

$$L_{\text{behavior}}(B)={\text{title()}, \text{title(t)}, \text{author()}, \text{author(p)}, \text{year()}, \text{year(y)}, \text{subject()}, \text{subject(s)}, \text{publisher()}, \text{publisher(p)}}$$

$$C_p(P)=[B] \quad C_b(P)={SI} \quad P_d(P)=\phi$$

$$L_{\text{attributes}}(P)={\text{location:String, chairperson:R, committee:[R], AcceptanceRate:Real, contents:[CP]}}$$

$$L_{\text{behavior}}(P)={\text{location()}, \text{location(l)}, \text{chairperson()}, \text{chairperson(p)}, \text{committee()}, \text{committee(p)}, \text{AcceptanceRate()}, \text{AcceptanceRate(i)}, \text{contents()}, \text{contents(c)}}$$

$$C_p(J)=[\text{OBJECT}] \quad C_b(J)={SI} \quad P_d(J)=\phi$$

$$L_{\text{attributes}}(J)={\text{title:String, EditorInChief:R, EditorialBoard:[R], subject:String, publisher:String, contents:[JP]}}$$

$$L_{\text{behavior}}(J)={\text{title()}, \text{title(t)}, \text{EditorInChief()}, \text{EditorInChief(e)}, \text{EditorialBoard()}, \text{EditorialBoard(p)}, \text{subject()}, \text{subject(s)}, \text{publisher()}, \text{publisher(p)}, \text{contents()}, \text{contents(c)}}$$

**Example 2:** [Brother class] Given DBJournals as a brother of class J with:  $P_d(\text{DBJournals})={J}$ . By definition, DBJournals has the following characteristics:

$$C_p(\text{DBJournals})=[ ] \quad C_b(\text{DBJournals})={ }$$

$$L_{\text{attributes}}(\text{DBJournals})={ } \quad L_{\text{behavior}}(\text{DBJournals})={\text{FindObjects()}}$$

$$L_{\text{instances}}(\text{DBJournals})={ }$$

Being its brother, DBJournals shares the characteristics of class J, and  $W_{\text{instances}}(\text{DBJournals})$  is determined by invoking the method  $\text{FindObjects}(\text{DBJournals})$ .

**Example 3:** [Virtual classes] Consider the following characteristics of some example derived classes.

**PP:** (PrintedPublications) includes all books and journals.

$$P_d(\text{PP})={B,J} \quad C_p(\text{PP})=[\text{OBJECT}] \quad C_b(\text{PP})={B, J}$$

$$W_{\text{instances}}(\text{PP})=W_{\text{instances}}(B) \cup W_{\text{instances}}(J) \quad L_{\text{instances}}(\text{PP})={ }$$

$$L_{\text{attributes}}(\text{PP})={\text{title:String, subject:String, publisher:String}}$$

$$L_{\text{behavior}}(\text{PP})={\text{title()}, \text{title(t)}, \text{subject()}, \text{subject(s)}, \text{publisher()}, \text{publisher(p)}, \text{FindObjects()}}$$

**NP:** (NewProceedings) includes only some characteristics from class P.

$$P_d(\text{NP})={P} \quad C_p(\text{NP})=[\text{OBJECT}] \quad C_b(\text{NP})={P}$$

$$W_{\text{instances}}(\text{NP})=W_{\text{instances}}(P) \quad L_{\text{instances}}(\text{NP})={ }$$

$$L_{\text{attributes}}(\text{NP})=W_{\text{attributes}}(B)Y{\text{location:String, chairperson:R}}$$

$$L_{\text{behavior}}(\text{NP})={W_{\text{behavior}}(B)Y{\text{location()}, \text{location(l)}, \text{chairperson()}, \text{chairperson(p)}, \text{FindObjects()}}}$$

**MRP:** includes all attributes of class RP and the name of the major researcher who contributed to the paper.

$$P_d(\text{MRP})={RP} \quad C_p(\text{MRP})=[RP] \quad C_b(\text{MRP})={ }$$

$$L_{\text{instances}}(\text{MRP})={ } \quad L_{\text{attributes}}(\text{MRP})={\text{name:String}}$$

$$W_{\text{instances}}(\text{MRP})=W_{\text{instances}}(RP) \quad L_{\text{behavior}}(\text{MRP})={\text{name()}, \text{name(s)}, \text{FindObjects()}}$$

## MAIN THRUST

Locally defined characteristics of class  $c$  include  $L_{\text{attributes}}(c)$  and  $L_{\text{behavior}}(c)$ . The inherited characteristics of class  $c$  are  $(W_{\text{attributes}}(c)-L_{\text{attributes}}(c))$  and  $(W_{\text{behavior}}(c)-L_{\text{behavior}}(c))$ . In this section, we present two algorithms to investigate adjusting for a given class: (1) its list of superclasses and (2) its set of subclasses. The first is necessary for classes in the third group, and the second is suitable for classes in the second group. However, both algorithms are required for classes in the fourth group.

Algorithm 1 investigates the possibility of maximizing reusability by adding some existing classes to  $C_p(c)$  or by pushing class  $c$  into the list of superclasses of some existing classes. We employ the following heuristic; having  $C_b(c)$  not empty helps in limiting our search to the direct and indirect superclasses of every class in  $C_b(c)$ . Otherwise, we have to consider and investigate all classes in the hierarchy, which is the general case when  $C_b(c)$  is empty. For each class  $c_b \in C_b(c)$ , Definition 1 is used to find the inheritance list of class  $c_b$ , denoted  $IL(c_b)$ , that includes all its direct and indirect superclasses. For the other case of having  $C_b(c)$  empty, the class hierarchy must be traced top-down and left to right in order to include all the classes present in the hierarchy inside  $IL(c)$ . Algorithm 1 checks for the possibility of including inside  $C_p(c)$  any of the classes in  $IL(c_b)$ , and the list of superclasses of the considered class is adjusted accordingly.

**Definition 1:** [Inheritance List] Given any class  $c$ ,  $IL(c)$  is defined using:

1. For every  $c_p \in C_p(c)$ ,  $c_p \in IL(c)$ ;
2. For every  $c' \in IL(c)$ , classes in  $C_p(c')$  are placed at the head of  $IL(c)$ ;
3. Nothing else belongs to  $IL(c)$ .

**Example 4:** [Inheritance List]  $IL(SI)=[B,J,P]$  and  $IL(P)=[B]$ .



**Algorithm 1:** [Maximize Inherited Characteristics]

**Input:** class  $c$  and  $C_b(c)$

**Output:** The class hierarchy adjusted to maximize reusability.

Let  $K$  be a list of classes, which is initially empty;

For every class  $c_b \in C_b(c)$  do

    Find  $IL(c_b)$  by Definition 1;

    For every  $c_j \in IL(c_b)$  do

#     if  $(W_{instances}(c) \subseteq W_{instances}(c_j))$  and  $(W_{behavior}(c) \subseteq W_{behavior}(c_j))$   
then

$C_p(c) = C_p(c) - C_p(c_j)$ ;

$C_p(c) = C_p(c) + \{c_j\}$ ;

    For every attribute  $iv \in L_{attributes}(c_j)$  do

        Let  $d_1$  be the underlying domain in

$L_{attributes}(c_j)$ ;

        Let  $d_2$  be the underlying domain in

$L_{attributes}(c)$ ; if  $d_1 = d_2$  then

$L_{attributes}(c) = L_{attributes}(c) - \{iv\}$ ;

    For every message  $m \in L_{behavior}(c_j)$  do

        Let  $m(c_m, f_m, r_m)$  be the header of the method underlying message  $m$  in  $L_{behavior}(c_j)$ ;

        Let  $m(c_n, f_n, r_n)$  be the header of the method underlying message  $m$  in  $L_{behavior}(c)$ ;

        if  $c_m = c_n$  and  $f_m = f_n$  and  $r_m = r_n$  then  
            Execute both methods using a sample set of objects from class  $c$ ;

        if both methods give the same result then

$L_{behavior}(c) = L_{behavior}(c) - \{m\}$ ;

        Else     Append  $c_j$  to list  $K$ ;

For every  $c_k \in C_p(c_b)$  do

    if  $c_k \in C_p(c)$  then      $C_p(c_b) = C_p(c_b) \cup \{c_k\}$ ;

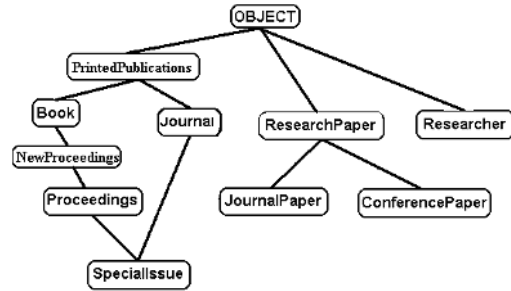
For every  $c_i \in K$  do

    Execute step #, by substituting  $c_i$  and  $c$  for  $c$  and  $c_j$ , respectively.

**Example 5:** [Pushing a class inside the hierarchy] Consider the two virtual classes PP and NP as input to Algorithm 1, which revises their characteristics to get:

**PP:**  $C_p(PP) = [\text{OBJECT}]$       $C_b(PP) = \{B, J\}$   
 $L_{instances}(PP) = \{\}$   
 $L_{attributes}(PP) = \{\text{title:String, subject:String, publisher:String}\}$   
 $L_{behavior}(PP) = \{\text{title}(), \text{title}(t), \text{subject}(), \text{subject}(s), \text{publisher}(), \text{publisher}(p), \text{FindObjects}()\}$

Figure 2. The revised class hierarchy after adding PP and NP



**NP:**  $C_p(NP) = [B]$       $C_b(NP) = \{\text{Proc.}\}$   
 $L_{instances}(NP) = \{\}$

$L_{attributes}(NP) = \{\text{location:String, chairperson:R}\}$   
 $L_{behavior}(NP) = \{\text{location}(), \text{location}(l), \text{chairperson}(), \text{chairperson}(p), \text{FindObjects}()\}$

As a result, the two virtual classes have been pushed into proper locations within the hierarchy as shown in Figure 2. Note that PP was not affected by Algorithm 1 because it was not violating reusability maximization. On the other hand, characteristics of NP have been modified, and NP has been pushed under B.

As illustrated in Example 5, Algorithm 1 is useful mainly for results of *projection*, as well as *union* and *difference* with non-brother<sup>2</sup> classes as operands. These results were proved to be superclasses of the operands (Alhadj & Arkun, 1993; Alhadj & Polat, 1994). But, it is also possible to have result  $vc$  as subclass of the operand(s); for example, the result of *nest* is a subclass of the first operand. It is semantically desirable to make the additional properties acquired by objects in  $W_{instances}(vc)$  reachable only when such objects are accessed from  $vc$  itself. Based on this, we developed a second approach where the new properties are made accessible solely from  $vc$ .

We introduced artificial classes into the hierarchy. Every class in the hierarchy produces an artificial class. Artificial classes contain only objects, and they inherit all other class characteristics. An artificial class  $ac$  contains objects that qualify to be in  $L_{instances}(c)$ , where  $c$  is the corresponding class in the hierarchy<sup>3</sup>; that is, objects migrate from  $L_{instances}(c)$  into  $L_{instances}(ac)$ , hence, are still considered in  $W_{instances}(c)$ . Further, given any two classes  $c'$  and  $c''$  that result in artificial subclasses  $ac'$  and  $ac''$ , respectively, having  $c'$  as a subclass of  $c''$  leads to have  $ac'$  as a subclass of  $ac''$ . It is an implementation issue to make artificial classes invisible and inaccessible to the user.

**Definition 2:** [Artificial Class] For every class  $c$ , there is a corresponding artificial class  $ac$  such that:

1.  $c \in IL(ac)$ , class  $ac$  is created first as a subclass of class  $c$ . Later on, as more artificial classes are created,  $ac$  may move down the hierarchy and becomes indirectly connected to class  $c$ ;
2. Any object  $oid$  that qualifies to be in  $L_{instances}(c)$  is added to  $L_{instances}(ac)$  instead. This way,  $oid \in L_{instances}(ac)$  and  $L_{instances}(ac) \subseteq W_{instances}(c)$ , because  $c \in IL(ac) \Rightarrow oid \in W_{instances}(c)$ ;
3. Given two classes  $c'$  and  $c''$  such that  $c' \in C_b(c'')$ , then  $ac' \in C_b(ac'')$ .

**Example 6:** [Artificial classes] Given two classes,  $c'$  and  $c''$  such that  $c' \in C_b(c'')$ , it is required to introduce a new class  $vc''$  as subclass of  $c''$  such that  $W_{instances}(c') \subseteq W_{instances}(vc'')$  and  $vc''$  has more characteristics (includes more attributes and behavior) than  $c'$ . We cannot add  $c'$  to  $C_b(vc'')$  as shown in Figure 3(a) because the additional characteristics of  $vc''$  will be seen from class  $c'$ , which is not the required target. We resolve this issue by following the artificial classes approach as shown in Figure 3(b).

From Figure 3(b), every class will continue to exist without being affected by class  $vc''$ , which will be able to recognize all related information in the hierarchy. Notice that artificial class  $ac''$  is not a direct subclass of  $c''$ . This is formalized next in Algorithm 2, where an artificial class  $ac$  is dropped from  $C_b(c)$  when  $ac$  becomes an indirect subclass of  $c$ .

**Algorithm 2:** [Subclass addition]

**Input:** Two classes  $c$  and  $vc$  such that  $vc$  is to be added to  $C_b(c)$

**Output:** The adjusted class hierarchy.

1.  $C_b(c) = C_b(c) \cup \{vc\}$
2.  $C_b(vc) = \{ac\}$
3. Construct artificial class  $avc$  according to Definition 2
4.  $C_b(c) = C_b(c) - \{ac\}$

To illustrate Algorithm 2, consider adding virtual class NRPs to the hierarchy shown in Figure 4, which is derived by adding artificial classes to the hierarchy shown in Figure 1. The obtained hierarchy is shown in Figure 5.

Finally, when an object  $oid$  is added to class  $c'$ , which may be either a direct or indirect subclass of class  $c''$ , it is necessary to consider the direct virtual subclasses of  $c''$  in order to polish  $oid$  such that it qualifies to be in  $W_{instances}(vc'')$  for every virtual class  $vc'' \in C_b(c'')$ . This is smoothly accomplished because  $oid$  is added to  $L_{instances}(ac')$  instead of  $L_{instances}(c')$ . Algorithm 2 forces  $ac'$  to be a subclass of  $vc''$ , and this leads to having  $L_{instances}(ac') \subseteq W_{instances}(vc'')$ . On adding  $oid$  to  $L_{instances}(ac')$ , it must carry all the characteristics defined for objects of the classes found in  $IL(ac')$ , but without violating inheritance rules. Since  $vc''$  is in  $IL(ac')$ ,  $oid$  will carry the characteristics defined for objects that qualify to be in  $W_{instances}(vc'')$ .

## FUTURE TRENDS

The derived type is treated as a separate entity in the work of Shaw and Zdonik (1990). In Carey, DeWitt, and Vandenberg (1988) new types created during query processing do not participate in inheritance in any way; neither do they become part of any inheritance hierarchy. The approach of Kaul, Drost, and Neuhold (1990) establishes only a local relationship between the derived type and the source type.

A new class in Kim (1989) is placed as a direct subclass of the root of the hierarchy. All the methods and attributes required to be in this new class have to be replicated from

Figure 3. Effect of adding  $vc''$  as subclass of  $c''$ : (a) without artificial classes; (b) with artificial classes

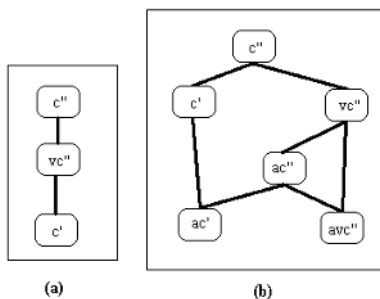


Figure 4. A class hierarchy with artificial classes

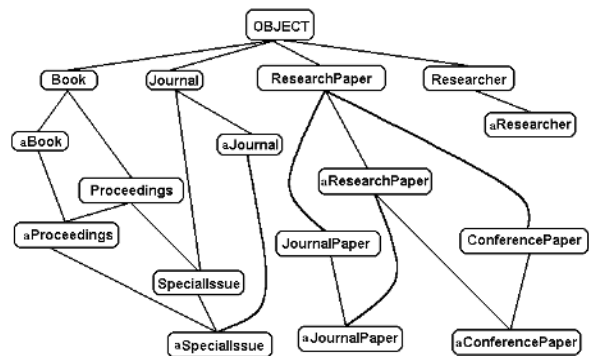
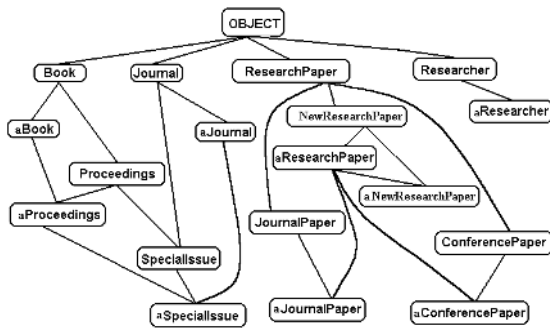


Figure 5. The class hierarchy in Figure 4 after adding virtual class NRP



the classes participated in the query. The work of Bertino et al. (1992) follows the approach proposed in Kim (1992) in having the result of a query made a direct subclass of the root; it does not have user-defined methods; and it much resembles a set of tuples in a relation. Views of XSQL (Kifer, Kim & Sagiv, 1992) are seen as sets of objects, which may serve as partial input for further queries. The user is expected to specify whether the result from a query is to be saved and how a new unique OID is to be calculated. The same approach of having the user explicitly classifying a query result is also used by  $O_2$  (Santos, Abiteboul & Delobel, 1994). There is no restriction on the placement, and the user may easily violate inheritance constraints by specifying the result to be a subclass of some other unrelated class. Such an approach puts too much overhead on the user side.

In OQL (Alashqur, Su & Lam, 1989), results obtained from queries are saved as new objects if save is specified by the user. The new objects exist outside the hierarchy and may be referred to only by using the name assigned by the user. Placing a newly derived subdatabase outside the hierarchy would ensure that no inheritance rules are violated. However, reusability is not achieved.

Some approaches employ views to handle schema changes. The work of Tresch and Scholl (1993) presents several examples of schema changes that can be simulated using views. However, implementation solutions and how to push a persistent view into its proper place within the hierarchy are not mentioned. Two other approaches that deal with schema changes are Bertino (1992) and Bratsberg (1992). The former does not consider the effect of schema changes on subclasses. The latter separates type and extent hierarchies, hence, violates the full inheritance invariant. Finally, the work of Ra and Rundensteiner (1997) is based on the MultiView system (Rundensteiner, 1992) with some extensions. However, it does not handle the automated proper placement of persistent views. Rather, the user has to specify the position. Finally, researchers recently real-

ized the importance of views, schema changes, and integration within the context of XML technologies (Kang & Lim, 2002; Pedersen & Pedersen, 2003). We argue that the approach proposed in this article for object-oriented databases may be easily adapted for XML. In fact, it is essential to facilitate for dynamic changes of the structure of the XML schema for several reasons, including correcting design errors in the schema, allowing expansion of the application scope over time, or accounting for the merging of several XML schemas into a single common application.

As a result, none of the already mentioned approaches handle the proper placement of persistent views in the hierarchy with the aim of maximizing reusability. This study of the existing approaches was the major motivation to deal with the problem more seriously and to find the solutions presented in this article. Further, the approach presented in this article may be easily adapted to XML.

## CONCLUSION

For a class to be properly placed inside the hierarchy, it should be defined by a user with complete knowledge about details of the class hierarchy. This is almost impossible in a dynamic environment where multi-users are accessing a common hierarchy with each user performing dynamic schema changes related to the user's specific domain of interest. Because of this, we automated the process and developed a system that can handle the proper placement of new classes in the hierarchy with minimum user interference. Naive and professional users benefit from our system regardless of their knowledge about the hierarchy. We concentrated on classes that correspond to persistent views; however, our system is still able to deal with other classes added to the hierarchy using class definition constructs. Each of the latter classes belongs to one of the four groups identified for virtual classes, depending on the ability of the user to specify related subclasses or superclasses. The worst case is having the user unable to specify direct subclasses and superclasses of a new class. Such a class is considered in the fourth group, and both its superclasses and subclasses are investigated. As a result of this study, once again we demonstrated that it is necessary for an object-oriented model to support multiple inheritance. Finally, researchers recently developed approaches to handle schema changes for XML. For instance, the work of Koeller and Rundensteiner (2002) handles schema-restructuring view maintenance operations as schema changes and vice versa. Kozankiewicz et al. (2002) present a new approach to virtual, updatable views for a query language addressing XML native data-

bases. Su et al. (2001) developed a framework for managing the evolution of DTDs and XML documents. Lerner (2000) developed a system that handles type changes by comparing schemas and then produces a transformer that can update data in a database to correspond to a newer version of the schema. Such approaches may benefit from the work described in this article.

## REFERENCES

- Alashqur, A., Su, S. Y., & Lam, H. (1989). OQL: A query language for manipulating object-oriented databases. *Proceedings of the VLDB*, Amsterdam, The Netherlands (pp. 433-442).
- Alhajj, R., & Arkun, M.E. (1993). A query model for object-oriented database systems. *Proceedings of the IEEE-ICDE*, Vienna, Austria.
- Alhajj, R., & Polat, F. (1994). Closure maintenance in an object-oriented query model. *Proceedings of the ACM-CIKM*, Maryland.
- Bertino, E. (1992). A view mechanism for object-oriented databases. *Proceedings of the EDBT*, Vienna, Austria.
- Bertino, E., et al. (1992). Object-oriented query languages: The notion and the issues. *IEEE TKDE*, 4(3), 223-237.
- Bratsberg, S.E. (1992). Unified class evolution by object-oriented views. *Proceedings of the ER. Lecture Notes in Computer Science*, Karlsruhe, Germany (pp. 423-439).
- Carey, M.J., DeWitt, D.J., & Vandenberg, S.L. (1988). A data model and a query language for EXODUS. *Proceedings of the ACM-SIGMOD* (pp. 413-423).
- Kang, H., & Lim, J. (2002). Deferred incremental refresh of XML materialized views. *Proceedings of the CAiSE* (pp. 742-746).
- Kaul, M., Drosten, K., & Neuhold, E.J. (1990). Viewsystem: Integrating heterogeneous information bases by object-oriented views. *Proceedings of the IEEE-ICDE*, Los Angeles, California.
- Kifer, M., Kim, W., & Sagiv, Y. (1992). Querying object-oriented databases. *Proceedings of the ACM-SIGMOD*, San Diego, California.
- Kim, W. (1989). A model of queries for object-oriented databases. *Proceedings of the VLDB* (pp. 423-432).
- Koeller, A., & Rundensteiner, E.A. (2002). Incremental maintenance of schema-restructuring views. *Proceedings of the EDBT*, Prague, Czech Republic (pp. 354-371).
- Kozankiewicz, H., et al. (2002). *Updateable object views* (Tech. Rep. No. 950). Institute of Computer Science of PAS.
- Lerner, B.S. (2000). A model for compound type changes encountered in schema evolution. *ACM TODS*, 25(1), 83-127.
- Pedersen, D., & Pedersen, T.B. (2003). Achieving adaptivity for OLAP-XML federations. *Proceedings of the ACM DOLAP* (pp. 25-32).
- Ra, Y.-G., & Rundensteiner, E.A. (1997). A transparent schema-evolution system based on object-oriented view technology. *IEEE TKDE*, 9(4), 600-624.
- Rundensteiner, E.A. (1992). MultiView: A methodology for supporting multiple views in object-oriented databases. *Proceedings of the VLDB*, Vancouver, Canada (pp. 187-198).
- Santos, C.S., Abiteboul, S., & Delobel, C. (1994). Virtual schemas and bases. *Proceedings of the EDBT*, Cambridge, UK (pp. 81-94).
- Shaw, G., & Zdonik, S. (1990). A query algebra for object-oriented databases. *Proceedings of the IEEE-ICDE* (pp. 154-162).
- Su, H., et al. (2001). XEM: Managing the evolution of XML documents. *Proceedings of the IEEE RIDE* (pp. 103-110).
- Tresch, M., & Scholl, M.H. (1993). Schema transformation without database reorganization. *SIGMOD Records*, 22, 21-27.

## KEY TERMS

**Base Class:** User-defined class; a collection of objects that have the same behavior and state definition.

**Class Hierarchy:** A directed acyclic graph (DAG) that describes the subclass/superclass relationships among classes. Each node represents a class, the children of a node represent the direct subclasses of a class, and the parents of a node represent the direct superclasses of a class.

**Inheritance:** The ability of a superclass to pass its characteristics (methods and instance variables) onto its subclasses, allowing subclasses to reuse these characteristics.

**Multiple Inheritance:** The capability of a class of objects to inherit attributes and behavior from more than one superclass.

**Object:** A data structure that encapsulates behavior (operations) and data (state).

**Subclass:** A class that is derived from at least one other class.

**Superclass:** A class from which a particular class is derived via inheritance.

**View or Virtual Class:** A class derived using a query expression.

**XML:** eXtensible Markup Language, a data format widely used for data exchange over the iInternet.

## ENDNOTES

- <sup>1</sup> In this article, the two terms, *views* and *virtual class*, will be used to refer to a *derived class*.
- <sup>2</sup> Operands become union-compatible when they are brother classes. Consequently, the union and difference operations can be transformed into a selection.
- <sup>3</sup> For every class *c*, the corresponding artificial class will have the same name prefixed with the letter “a”.



# Querical Data Networks

**Cyrus Shahabi**

*University of Southern California, USA*

**Farnoush Banaei-Kashani**

*University of Southern California, USA*

## INTRODUCTION

Recently, a family of massive self-organizing data networks has emerged. These networks mainly serve as large-scale distributed query-processing systems. We term these networks *querical data networks (QDN)*. A QDN is a federation of a dynamic set of peer, autonomous nodes communicating through a transient-form interconnection. Data is naturally distributed among the QDN nodes in extra-fine grain, where a few data items are dynamically created, collected, and/or stored at each node. Therefore, the network scales linearly to the size of the data set. With a dynamic data set, a dynamic and large set of nodes, and a transient-form communication infrastructure, QDNs should be considered as the new generation of distributed database systems with significantly less constraining assumptions as compared to their ancestors. Peer-to-peer networks (Daswani, Garcia-Molina, & Yang, 2003) and sensor networks (Akyildiz, Su, Sankarasubramaniam, & Cayirci, 2002; Estrin, Govindan, Heidemann, & Kumar, 1999) are well-known examples of QDNs.

QDNs can be categorized as instances of “complex systems” (Bar-Yam, 1997) and studied using the complex system theory. Complex systems are (mostly natural) systems hard (or complex) to describe information-theoretically and hard to analyze computationally. QDNs share the same characteristics with complex systems and, particularly, bear a significant similarity to a dominating subset of complex systems most properly modeled as large-scale interconnection of functionally similar (or peer) entities. The links in the model represent some kind of system-specific entity-to-entity interaction. Social networks, a network of interacting people, and cellular networks, a network of interacting cells, are two instances of such complex systems. With these systems, complex global system behavior (e.g., a social revolution in a society, or food digestion in a stomach!) is an emergent phenomenon, emerging from simple local interactions. Various fields of study, such as sociology, physics, biology, chemistry, etc., were founded to study different types of initially simple systems and have been gradually matured to analyze and describe

instances of incrementally more complex systems. An interdisciplinary field of study, the complex system theory,<sup>1</sup> was recently founded based on the observation that analytical and experimental concepts, tools, techniques, and models developed to study an instance of complex system at one field can be adopted, often almost unchanged, to study other complex systems in other fields of study. More importantly, the complex system theory can be considered as a unifying metatheory that explains common characteristics of complex systems. One can extend application of the complex system theory to QDNs by:

1. Adopting models and techniques from a number of impressively similar complex systems to design and analyze QDNs as an instance of engineered complex systems; and
2. Exporting the findings from the study of QDNs (which are engineered, hence, more controllable) to other complex system studies.

This article is organized in two parts. In the first part, we provide an overview, where we (1) define and characterize QDNs as a new family of data networks with common characteristics and applications, and (2) review possible database-like architectures for QDNs as query processing systems and enumerate the most important QDN design principles. In the second part of the article, as the first step toward realizing the vision of QDNs as complex distributed query-processing systems, we focus on a specific problem, namely, the problem of effective data location (or search) for efficient query processing in QDNs. We briefly explain two parallel approaches, both based on techniques/models borrowed from the complex system theory, to address this problem.

## BACKGROUND

Here, we enumerate the main componental characteristics and application features of a QDN.

## Componental Characteristics

A network is an interconnection of nodes via links, usually modeled as a graph. Nodes of a QDN are often massive in number and bear the following characteristics:

- **Peer functionality:** All nodes are capable of performing a restricted but similar set of tasks in interaction with their peers and the environment, although they might be heterogeneous in terms of their physical resources. For example, joining the network and forwarding search queries are among the essential peer tasks of every node in a peer-to-peer network.
- **Autonomy:** Aside from the peer tasks mentioned above, QDN nodes are autonomous in their behavior. Nodes are either self-governing or governed by out-of-control uncertainties. Therefore, to be efficacious and applicable, the QDN engineering should avoid imposing requirements to and making assumptions about the QDN nodes.<sup>2</sup> For example, strict regulation of connectivity (e.g., enforcing number of connections and/or target of connections) might be an undesirable feature for a QDN design.
- **Intermittent presence:** Nodes may frequently join and leave the network based on their autonomous decision, due to failures, etc.

On the other hand, links in various QDNs stand for different forms of interaction and communication. Links may be physical or logical, and they are fairly inexpensive to rewire. Therefore, a QDN is a large-scale federation of a dynamic set of autonomous peer nodes building a transient-form interconnection. Conventional approaches developed to model and analyze traditional distributed database systems (and classical networks, as their underlying communication infrastructure) are either too weak (oversimplifying) or too complicated (overcomplicated) to be effective with large-scale and topology-transient QDNs. The complex system theory (Bar-Yam, 1997), on the other hand, provides a set of conceptual, experimental, and analytical tools to contemplate, measure, and analyze systems such as QDNs.

## Application Features

A QDN is applied as a distributed source of data (a *data network*) with nodes that are specialized for cooperative query processing and data retrieval. The node cooperation can be as trivial as forwarding the queries or as complicated as in-network data analysis. In order to

enable such an application, QDN should support the following features:

- **Data-centric naming, addressing, routing, and storage:** With a QDN, queries are declarative; i.e., query refers to the names of data items and is independent of the location of the data. The data may be replicated and located anywhere in the data network, the data holders are unknown to the querier and are only intermittently present, and the querier is interested in data itself rather than the location of the data. Therefore, naturally QDN nodes should be named and addressed by their data content rather than an identifier in a virtual name space such as the IP address space. Consequently, with data-centric naming and addressing of the QDN nodes (Heidemann et al., 2001), routing (Ratnasamy, Francis, Handley, Karp, & Shenker, 2001) and storage (Ratnasamy et al., 2003) in QDN are also based on the content. It is interesting to note that non-procedural query languages such as SQL also support declarative queries and are appropriate for querying data-centric QDNs.
- **Self-organization for efficient query processing:** QDNs should be organized for efficient query processing. A QDN can be considered as a database system with the data network as the database itself (see the next section). QDN nodes cooperate in processing the queries by retrieving, communicating, and preferably on-the-fly processing of the data distributed across the data network. To achieve efficiency in query processing with high resource utilization and good performance (e.g., response time, query throughput, etc.), QDNs should be *organized* appropriately. Examples of organization are: intelligent partitioning of the query to a set of sub-queries to enable parallel processing, or collaborative maintenance of the data catalogue across the QDN nodes. However, the peer tasks of the QDN nodes should be defined such that they *self-organize* to the appropriate organization. In other words, organization must be a collective behavior that emerges from local interactions among nodes; otherwise, the dynamic nature and large scale of a QDN render any centralized micromanagement of a QDN unscalable and impractical.

## VISION: A DATABASE QUERYING FRAMEWORK FOR QDNS

In the previous section, we defined a querical data network (QDN) as a distributed data source and a query

processing system. On the other hand, a database system (DBS) is designed specifically as a general framework for *convenient* and *efficient* querying of static or dynamic collections of interrelated data. Thus, querying QDNs can be designed, developed, and executed by adopting a DBS as the general-purpose querying framework, leveraging on its rich abstractions, theories, and processing methods (Govindan et al., 2002; Harren et al., 2002). In particular, adopting the DBS framework potentially results in (1) convenient and rapid application development for users at the conceptual level, by providing well-known and transparent abstractions independent of the implementation of the querying, and (2) efficient query processing at the physical level, by providing a general-purpose querying component that adopts and customizes query processing methods from the database literature as well as other related fields such as distributed computing.

Here, we depict and compare potential architectures for the DBS framework, define a taxonomy of approaches to generalize this querying framework for the entire family of QDNs, and enumerate some important design principles for query processing in QDNs.

## Architecture

The DBS querying framework for QDNs suggests a two-level architecture consisting of a conceptual level and a physical level. At the conceptual level, queries are defined based on the conceptual schema of the data network, independent of the physical implementation of the query processing. The physical data independence allows rapid development of QDN applications, similar to the database application development. For instance, a peer-to-peer application that monitors violation of speed limit at a highway can pose the following query to a (hypothetical) mobile peer-to-peer network of vehicles:

```
SELECT      vehicle-ID
FROM        Cars
WHERE       (speed > 70) AND (location IN
            "Highway No. 88")
```

Similarly, a heat alert application may pose the following query to a heat detection sensor field: *“Report the outlier heat data in all offices during night.”*

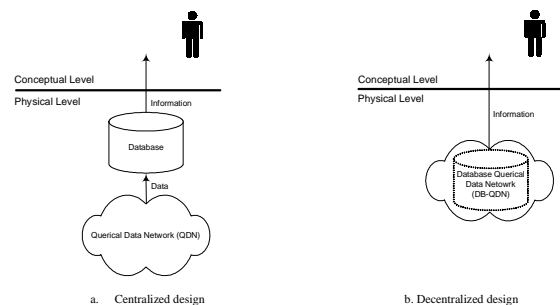
Queries are executed at the physical level. There are two extreme choices of design for query processing at the physical level: *centralized* and *decentralized*; a hybrid design may also be meaningful for particular applications. With a centralized design, a potential querier (i.e., one of the QDN nodes or an outsider) receives the query from the QDN application at the conceptual level. The query is disseminated to all QDN nodes, where the

query is interpreted based on the local conceptual schema, and all raw data required to process the query are transmitted back to the querier via the network (see Figure 1a). The querier treats the collected data as a centralized database and processes and analyzes the data to respond to the query. With this scheme, data sourcing and query processing are completely decoupled; the data network maintains and communicates the data, and the querier processes the query individually. The cooperation among QDN nodes is limited to forwarding the query and the raw data, and the network is used only as a point-to-point communication infrastructure to communicate the data between the sources and the querier.

A centralized design is simple to implement. The centralized query processing approach is also resource-efficient for trivial queries such as typical peer-to-peer search queries. However, with complex queries, where, for example, two 10,000-record tables must be joined to retrieve a few records, the overhead of transmitting the entire content of the tables to the querier for central processing is overwhelming and renders the centralized approach impractical. With most QDNs (e.g., sensor networks), this overhead is particularly intolerable due to the several orders of magnitude higher cost of communication as compared with that of computation in typical QDN nodes. Instead, in-network and on-the-fly processing of the query potentially eliminates the redundant communication of the data; hence, it is more efficient and scalable. The decentralized design adopts the latter approach.

With the decentralized design, the QDN is itself both the source of the data and the query-processing unit (see Figure 1b). The data sourcing/communication and data analysis tasks are integrated, and QDN nodes cooperate to perform both tasks within the network. With this approach, queries are processed based on the following general scheme: Query is disseminated to a selected set of QDN nodes; QDN nodes exploit their

Figure 1. Database system framework for querying querical data networks (QDNs)



computing power to process the query locally, cooperatively, in parallel, and in a distributed fashion, to extract the required information from the raw data; and eventually, the extracted information merges to comprise the final query result while traveling toward the querier through the network.

The efficiency of the decentralized design is due to in-network processing. With this approach, communication of the raw data is restricted to short-range (hence, less costly) communications among local cooperative-analysis groups of QDN nodes, which process the voluminous raw data and extract the concise required information to respond to the query. Although the decentralized design potentially promises an efficient and scalable querying framework for QDNs, realizing this design is a challenging endeavor and requires designing efficient distributed and cooperative query-processing mechanisms that comply with specific characteristics of QDNs.

## Taxonomy

Based on the two fundamentally distinct design choices for the physical level of the DBS framework (i.e., centralized and decentralized), one can recognize two approaches to implement a DBS-based querying system for QDNs:

1. **Database for QDN:** This approach corresponds to the querying systems with centralized query processing. These systems are similar to other centralized database applications, where data are collected from some data sources (depending on the host application, the data sources can be text documents, media files, and, in this case, QDN data) to be separately and centrally processed.
2. **Database-QDN (DB-QDN):** The systems that are designed based on this approach strive to implement query processing in a decentralized fashion within the QDN; hence, in these systems “*QDN is the database.*”

## Design Principles

By definition QDNs tend to be large-scale systems and their potential benefits increase as they grow in size. Therefore, between the two types of DBS-based querying systems for QDNs, the database-QDNs (DB-QDNs) are more promising because they are scalable and efficient. Among the most important design principles for distributed query processing at DB-QDNs, one can distinguish the following:

1. **In-network query processing:** In-network query processing is the main distinction of DB-QDNs. In-network query processing techniques should be implemented in a distributed fashion, ensuring minimal communication overhead and optimal load-balance.
2. **Transaction processing with relaxed properties:** Due to the dynamic nature of QDNs, requiring ACID-like properties for transaction processing in DB-QDNs is too costly to be practical and severely limits the scalability of such processing technique. Hence, transaction-processing properties should be relaxed for DB-QDNs.
3. **Adaptive query optimization:** Since QDNs are inherently dynamic structures, optimizing query plans for distributed query execution in DB-QDNs should also be a dynamic/adaptive process. Adaptive query optimization techniques are previously studied in the context of central query processing systems (Avnur & Hellerstein, 2000).
4. **Progressive query processing:** Distributed query processing tends to be time-consuming. With real-time queries, the user may prefer receiving a rough estimation of the query result quickly rather than waiting long for the final result. The rough estimation progressively enhances to the accurate and final result. This approach, termed *progressive* query processing (Schmidt & Shahabi, 2002a), allows users to rapidly obtain a general understanding of the result, to observe the progress of the query, and to control its execution (e.g., by modifying the selection condition of the query) on the fly.
5. **Approximate query processing:** Approximation techniques such as *wavelet-based* query processing can effectively decrease the cost of the query while producing highly accurate results (Schmidt & Shahabi, 2002b). Inherent uncertainty of the QDN data, together with the relaxation of the query semantics, justifies application of approximation techniques to achieve efficiency.

## FUTURE TRENDS

One of the most fundamental functionalities required to realize a DB-QDN is the *search* primitive. Efficient location of the data within the QDN, a large-scale and dynamic system with a distributed and dynamic data set, is a challenging task vital to QDN query processing. For the remainder of this section, we briefly explain two parallel approaches one can adopt from the complex system theory to address the QDN search problem. First, we discuss a self-organizing mechanism to structure



the topology of the QDN to a search-efficient topology. This topology can be considered as a distributed index structure that organizes the nodes and, therefore, the data content of the nodes for efficient search. For the design of the search-efficient QDN topology as well as the search dynamics, we are inspired by the “small world” models. Small worlds are models proposed to explain efficient communication in a social network, which is a semi-structured complex system.

Second, we propose an efficient query flooding mechanism for QDNs. Flooding is not only required for broadcast queries at all QDNs but also for uni-cast and multi-cast queries in unstructurable/unindexible QDNs. With these QDNs, the extreme dynamism of the QDN topology and the extreme autonomy of the QDN nodes render any attempt to impose even a semi-structure on the network by an index-like structure inefficient and/or impossible. We use percolation theory, an analytical tool borrowed from the complex system theory, to formalize and analyze such efficient flooding mechanism.

### Probabilistic Indexing of QDNs for Efficient Approximate Query Processing

Considering a QDN as a database (with every node of the QDN as the potential entry point of the query), similar to traditional databases QDN should be “indexed” for efficient processing of the queries. To process approximate queries,<sup>3</sup> we propose self-organizing the interconnection of the QDN based on the data content of the QDN nodes. With this organization, the network distance between every two nodes is positively correlated with the similarity of their data content with high probability. This approach results in an indexed network with distinguishable data localities, allowing efficient routing of the queries toward the nodes holding the result set of the query. The similarity measurements performed by each node while joining the QDN to select an appropriate set of neighbors can be thought of as the off-line pre-computations required to create the index for efficient online query processing. Also, the topology of the generated interconnection should be compared with the tree-like topologies of the traditional hierarchical index structures in centralized databases. In addition to allowing efficient navigation/traversal of the data set, this topology should support the dynamism of the data set and the network and, more importantly, should avoid assuming a central entry point (the root node in hierarchical indices) for the query, in order to balance the query load among all the nodes of the QDN.

It turns out that a probabilistic “small world” model, which is a topology proposed to explain efficient communication in social networks, is a perfect candidate topology to index QDNs. With our *searchable* QDN model (Banaei-Kashani & Shahabi, 2003b), we propose a self-organization mechanism that generates a QDN with small-world topology based on a recently developed small-world model (Watts, Dodds, & Newman, 2002). We complement the generated small-world network topology (i.e., the index) with a query forwarding mechanism (i.e., the index lookup technique) that effectively routes partial-match queries toward the QDN nodes that store the matching data items. Currently, we are focusing on extending this query routing technique to support more challenging queries such as range queries and nearest-neighbor queries.

### Criticality-Based Probabilistic Flooding

Flooding is a common mechanism used in many networks, including QDNs, to broadcast a piece of information (e.g., an alert or a search query) from a source node to other nodes of the network. With normal flooding, each node always forwards the received information to all its neighbors (i.e., directly connected nodes). In spite of many beneficial features, such as providing broad coverage and guaranteeing minimum delay, normal flooding is not a scalable communication mechanism, mainly because of the communication overhead it imposes to the system. To alleviate this problem, we introduce *probabilistic flooding* (Banaei-Kashani & Shahabi, 2003a). With probabilistic flooding, unlike normal flooding, a node forwards the information to its neighbor probabilistically, with probability  $p$ . By changing the probability value  $p$ , we can control the effective connectivity of the network while information is forwarded. The idea is to tune the probability value  $p$  to a critical operation point (the phase transition point) such that statistically the network remains connected (to preserve full reachability) while redundant paths are eliminated. Percolation theory (Stauffer & Aharony, 1992) is an analytical tool from the complex system theory extensively used to study probabilistic diffusion-like physical phenomena; e.g., diffusion of oil inside porous rocks in oil reservoir, a physical complex system. We use percolation theory to formalize the probabilistic flooding approach as a query-diffusion problem and to find its critical (optimal) operating point rigorously. Our formal analysis shows that the critical value of  $p$  can be as low as 1%, which translates to 99% reduction in communication overhead of flooding, hence, scalable flooding.



## CONCLUSION

In this article, we identified querical data networks (QDNs) as a family of data networks recently emerging as a new generation of distributed database systems with significantly less constraining assumptions. We envision a QDN as a distributed query processing system with a database-like architecture. In search of an effective approach to design and analyze database-QDNs, we find the complex system theory, a theory that explains a family of systems with characteristics that bear significant similarity to those of QDNs, extremely helpful. As an instance application of this approach, we provide two parallel solutions for the QDN search problem inspired by models adopted from the complex system theory.

## ACKNOWLEDGMENTS

This research has been funded in part by NSF grants EEC-9529152 (IMSC ERC), IIS-0082826 (ITR), IIS-0238560 (CAREER), IIS-0324955 (ITR), and IIS-0307908 and unrestricted cash gifts from Okawa Foundation and Microsoft. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

## REFERENCES

- Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002). A survey on sensor networks. *IEEE Communications Magazine*, 40(8), 102-114.
- Avnur, R., & Hellerstein, J. (2000). Eddies: Continuously adaptive query processing. *Proceedings of ACM International Conference on Management of Data* (pp. 261-272).
- Banaei-Kashani, F., & Shahabi, C. (2003a). Criticality-based analysis and design of unstructured peer-to-peer networks as complex systems. *Proceedings of the Third International Workshop on Global and Peer-to-Peer Computing (GP2PC) in conjunction with CCGrid* (pp. 351-359).
- Banaei-Kashani, F., & Shahabi, C. (2003b). Searchable querical data networks. In *Lecture notes in computer science: Vol. 2944* (pp. 17-32). Berlin, Germany: Springer-Verlag.
- Bar-Yam, Y. (1997). *Dynamics of complex systems*. New York: Westview Press.
- Daswani, N., Garcia-Molina, H., & Yang, B. (2003). Open problems in data-sharing peer-to-peer systems. *Proceedings of the ninth International Conference on Database Theory* (pp. 1-15).
- Estrin, D., Govindan, R., Heidemann, J., & Kumar, S. (1999). Next century challenges: Scalable coordination in sensor networks. *Proceedings of International Conference on Mobile Computing and Networks* (pp. 256-262).
- Govindan, R., Hellerstein, J., Hong, W., Madden, S., Franklin, M., & Shenker, S. (2002). *The sensor network as a database* (Tech. Rep. No. 02-771). University of Southern California, Los Angeles, CA.
- Harren, M., Hellerstein, J., Huebsch, R., Loo, B. T., Shenker, S., & Stoica, I. (2002). Complex queries in DHT-based peer-to-peer networks. *Proceedings of the first International Workshop on Peer-to-Peer Systems*.
- Heidemann, J., Silva, F., Intanagonwiwat, C., Govindan, R., Estrin, D., & Ganesan, D. (2001). Building efficient wireless sensor networks with low-level naming. *Proceedings of the Symposium on Operating Systems Principles* (pp. 146-159).
- Ratnasamy, S., Francis, P., Handley, M., Karp, R., & Shenker, S. (2001). A scalable content addressable network. *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (pp. 161-172).
- Ratnasamy, S., Karp, B., Shenker, S., Estrin, D., Govindan, R., Yin, L., & Yu, F. (2003). Data-centric storage in sensornets with GHT, a geographic hash table. *Mobile Networks and Applications*, 8(4), 427-442.
- Schmidt, R., & Shahabi, C. (2002a). How to evaluate multiple range-sum queries progressively. *21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (pp. 133-141).
- Schmidt, R., & Shahabi, C. (2002b). Propolyne: A fast wavelet-based algorithm for progressive evaluation of polynomial range-sum queries. *Eighth Conference on Extending Database Technology* (pp. 664-681).
- Stauffer, D., & Aharony, A. (1992). *Introduction to percolation theory* (2nd ed.). London: Taylor and Francis.
- Watts, D. J., Dodds, P. S., & Newman, M. E. J. (2002). Identity and search in social networks. *Science*, 296, 1302-1305.

## KEY TERMS

**Complex Systems:** Complex systems is a new field of science studying how parts of a complex system give rise to the collective behaviors of the system. Complexity (information-theoretical and computational) and emergence of collective behavior are the two main characteristics of such complex systems. Social systems formed (in part) out of people, the brain formed out of neurons, molecules formed out of atoms, and the weather formed out of air flows are all examples of complex systems. The field of complex systems cuts across all traditional disciplines of science, as well as engineering, management, and medicine.

**Content-Centric Networks:** A content-centric network is a network where various functionalities such as naming, addressing, routing, storage, etc. are designed based on the content. This is in contrast with classical networks that are node-centric.

**Distributed Hash Tables (DHTs):** A distributed index structure with hash table-like functionality for information location in the Internet-scale distributed computing environments. Given a key from a pre-specified flat identifier space, the DHT computes (in a distributed fashion) and returns the location of the node that stores the key.

**Peer-to-Peer (P2P) Networks:** A peer-to-peer network is a distributed, self-organized federation of *peer* entities, where the system entities collaborate by sharing resources and performing cooperative tasks for mutual benefit. It is often assumed that such a federation lives, changes, and expands independent of any distinct service facility with global authority.

**Percolation Theory:** Assume a grid of nodes where each node is occupied with probability  $p$  and empty with probability  $(1-p)$ . Percolation theory is a quantitative (statistical-theoretical) and conceptual model for under-

standing and analyzing the statistical properties (e.g., size, diameter, shape, etc.) of the clusters of occupied nodes as the value of  $p$  changes. Many concepts associated with complex systems such as clustering, fractals, diffusion, and particularly phase transitions are modeled as the percolation problem. The significance of the percolation model is that many different problems can be mapped to the percolation problem; e.g., forest-fire spread, oil field density estimation, diffusion in disordered media, etc.

**Sensor Networks:** A sensor network is a network of low-power, small form-factor sensing devices that are embedded in a physical environment and coordinate amongst themselves to achieve a larger sensing task.

**Small-World Models:** It is believed that almost any pair of people in the world can be connected to one another by a short chain of intermediate acquaintances, of typical length about six. This phenomenon is colloquially referred to as the “six degrees of separation,” or, equivalently, the “small world” effect. Sociologists propose a number of topological network models, the small-world models, for the social network to explain this phenomenon.

## ENDNOTES

- <sup>1</sup> Go to New England Complex Systems Institute (<http://necsi.org/>) for more information about the complex system theory.
- <sup>2</sup> One can consider peer tasks as *rules of federation*, which govern the QDN but do not violate autonomy of individual nodes.
- <sup>3</sup> Considering the transience of the QDN structure and the dynamism of the data set, *exact* query processing with zero false dismissal is not a practical option.

# Query Operators in Temporal XML Databases

Kjetil Nørnvåg

Norwegian University of Science and Technology, Norway

## INTRODUCTION

The amount of data available in XML is rapidly increasing and at the same time the price of mass storage is rapidly decreasing, and this makes it possible to store larger amounts of data. The contents of a database or data warehouse are seldom static. New documents are created, documents are deleted and, more important, documents are updated. In many cases, one wants to be able to search in historical (old) versions, retrieve documents that were valid at a certain time, query changes to documents, and so forth. (Note that although this process is somewhat similar to general document versioning maintenance, the aspect of time makes possibilities and appropriate solutions different.) The “easiest” way to do this is to store all versions of all documents in the database and use a middleware layer to convert temporal query language statements into conventional statements, executed by an underlying database system (an example of such a system is TeXOR; Nørnvåg, Limstrand, & Myklebust, 2003). Although this approach makes the introduction of temporal support easier, it can be difficult to achieve good performance: temporal query processing is in general costly, and the cost of storing the complete document versions can be high. Thus, a temporal XML database system is necessary.

In our group we have developed several database systems suitable for storing XML data, including V2 (Nørnvåg, 2003b). An important issue is efficient query processing. To achieve this, appropriate query operators are needed. In this article, we describe the query operators necessary to execute such queries.

## BACKGROUND

### Related Work

A model for representing changes in semistructured data (i.e., DOEM) and a language for querying changes (i.e., Chorel) were presented by Chawathe, Abiteboul, and Widom (1998, 1999). Chorel queries are translated to Lorel (a language for querying semistructured data) and can therefore be viewed as a stratum approach.

An approach that is orthogonal, but still related to the work presented in this article, is to introduce valid time

features into XML documents, as presented by Grandi and Mandreoli (1999, 2000).

Many algorithms for execution of temporal query operators are based on the availability of temporal full-text indexes (Nørnvåg, 2002) that can support temporal text-containment queries. A number of such indexes have been proposed (e.g., see Nørnvåg, 2004, 2003a).

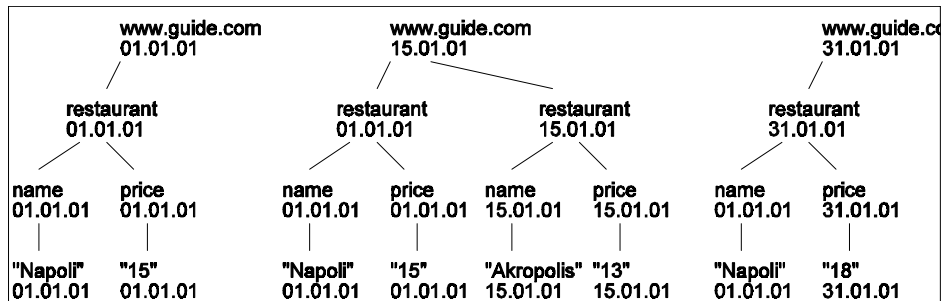
### Identity of Elements in Versioned XML Documents

An important issue that poses some additional difficulties in the context of XML and, in particular, in the context of XML documents retrieved from the Web (such as from an XML data warehouse), is identity of elements.

XML documents have a quasi-persistent identifier: the URL (*quasi*-persistent because documents on the Web frequently are moved). However, in general the elements of a document do not have an identity of their own that persists from one version of a document to the next, which implies that many queries can be difficult to express as well as expensive to execute. Two simple examples are (a) a query for the create-time of elements, and (b) a query asking for the previous version of a certain element. Thus, although elements seen from “the outside” do not have persistent identifiers, the storage system should support this feature to make it a part of the data model for the query language. One particular system that provides such functionality is Xyleme (Marian, Abiteboul, Cobena, & Mignet, 2001). The persistent identifiers in Xyleme, called XIDs, identify an element in a particular document in a time-independent manner and will not be reused when an element is deleted. The element ID (EID) is the concatenation of document ID and XID. Thus, an EID uniquely identifies a particular element in a particular document.

In a temporal XML database there will, in general, be more than one version of each element (i.e., different versions of an element have the same EID). In order to uniquely identify a particular version of an element, the timestamp can be used together with the EID. The identifier of a particular version of an element is the temporal EID (TEID) (i.e., the concatenation of EID and timestamp).

Figure 1. The restaurant list as retrieved on January 1, January 15, and January 31. The timestamps on each element is the time of update of the element or one of its children.



### Assumed Data Model

Documents in the database are often viewed as a forest of trees. Queries for certain versions of a document (or several documents) are similar to queries for a general set of XML documents, which can also be view as a forest of trees, or the forest of trees resulting from prefiltering (i.e., returning subtrees of documents, possibly more than one tree for each document).

Assume that:

- every element has a timestamp.
- every update of an element also implies update of the element it is contained in. Note that even if this *logically* has to be applied recursively up the document to the root, *it does not have to be implemented in this way.*

Note that the distinction between document timestamp and element timestamp is not significant for snapshot queries, only for change-oriented queries. An example of document versions can be seen in Figure 1. The document versions are versions of a restaurant guide database, as described in Chawathe et al. (1999). The restaurant guide will also be used in the examples below.

Note that in the physical storage model, it is unlikely that all versions of all documents are stored as complete versions. Instead, previous versions are stored as, for example, delta versions. In order to reconstruct these previous versions, we might have to retrieve and process the complete last versions as well as a number of delta documents.

### Examples of Temporal XML Queries

The main purpose of this section is to describe the kinds of queries that can be expected in a temporal XML

database (note that the query operators and associated algorithms are in general independent of which query language is actually used). The query language is based on a mix of Lorel, the Xyleme query language (Aguilera, Cluet, Veltri, Vodislav, & Watez, 2000), and elements of XPath and XQuery (World Wide Web Consortium, 2001). Note that even if Xyleme has support for historical versions or deltas, there is no special support for them in the query language. For example, a query returning all restaurants with prices less than \$10 could be written as:

```
SELECT R
FROM doc("http://guide.com"/)/restaurant R
WHERE R/price < 10
```

Assume that the results of an “outer query” are delivered as default in a document with enclosing tags named <results>. Each result from the SELECT expression is delivered in one element with tags named <result>. In most of the following example queries, complete paths are used (i.e., not containing a // operator). However, when querying semistructured data such as XML, many queries can be expected to contain the // operator.

To retrieve documents that are valid at a particular time (i.e., snapshot query), a timestamp is given for the path in the FROM clause, filtering out only those element versions valid at the particular time:

```
SELECT R
FROM doc("http://guide.com"/)/restaurant[26/01/2001] R
```

For more complex queries, or when more than one version to be selected is wanted, use the keyword EVERY instead of timestamp. For example, to retrieve the price history of the restaurant named Napoli:



```
SELECT TIME(R), R/price
FROM doc("http://guide.com/")/restaurant[EVERY] R
WHERE R/name = "Napoli"
```

where `TIME(R)` returns the timestamp of the element `R`. Further predicates on time can be included in the `SELECT` clause, including the delete and create time of elements. In order to query time relative to another time, for example `[NOW]` (which denotes the “current time”) or a certain time `DD/MM/YYYY`, expressions such as `NOW -14 DAYS` or `26/01/2001 -2 WEEKS` can be used.

## ALGEBRA OPERATORS

Operators needed in temporal XML query processing are now considered and are described in terms of input, output, and operation. In addition to the operators described here, the availability of traditional operators is assumed (e.g., projection and join) but are not discussed further.

Two of the operators are extensions of the `PatternScan` operator described in Aguilera et al. (2000). The `PatternScan` operator takes as input a forest of trees (which can be a set of EIDs) that are XML documents or filtered elements (subtrees) from an XML document, and a pattern tree that each tree will be matched against. The pattern tree includes information on projection as well as *isParentOf* and *isAscendantOf* relationships. Informally, we can define the operator as `PatternScan(F,pattern)`, where `F` is a forest of trees and `pattern` is the pattern tree. For more details on the `PatternScan` operator, see Aguilera et al. (2000).

## Overview of the Operators

The temporal query operators to be added are as follows:

- **TPatternScan(F,pattern,T)**: This is a temporal snapshot `PatternScan` operator. `TPatternScan` is similar to the `PatternScan` operator in Aguilera et al. (2000), except that it operates on the snapshot of documents valid at time `T`. The output of the operator is a set of TEIDs.
- **TPatternScanAll(C,pattern)**: `TPatternScanAll` returns all matches for `pattern` for all versions of the documents in a collection `C`. The output of the operator is a set of TEIDs.
- **DocHistory(document,TS,TE)**: Returns all the versions of a certain document valid in the interval `[TS,TE>`, where `[TS,TE>` is short for the time interval from `TS` to `TE`, including `TS` but not `TE` (open-ended upper bound). The output of the op-

erator is a set of TEIDs, where the TEIDs are roots of documents.

- **ElementHistory(EID,TS,TE)**: Returns all versions of an element valid in the interval `[TS,TE>`. The output of the operator is a set of TEIDs (with the EID in the TEID equal to the EID in the input parameters).
- **CreTime(TEID)**: Returns the create time of an element. This is useful for retrieving elements created before or after a particular time, for example, as in:

```
SELECT R
FROM ...
WHERE CREATE TIME(R)>=11/01/2001.
```

Note that EIDs are unique, so that when an element is deleted, the EID will not be reused for a new element. Thus, we do not strictly need timestamps in the element identification for the `CreTime` and `DelTime` operators. However, as shown in the expanded version of this article (Nørnvåg, 2002), the availability of timestamp can improve performance. The timestamps will in general be available in any case so that no extra cost is involved by assuming its availability.

- **DelTime(TEID)**: `DelTime` returns the delete time of an element.
- **PreviousTS(TEID)**
- **NextTS(TEID)**
- **CurrentTS(EID)**

Returns the timestamp of the previous/next/current version of a given element (note that timestamp is not needed for the current version, as this is given implicitly). The timestamp, together with the EID (i.e., the TEID), can be used for retrieving the version itself. These operators can be used in constructions such as

```
SELECT DISTINCT CURRENT(R)/name
FROM ...
WHERE ...,
```

which retrieves the current versions of elements (possibly generated from a temporal snapshot), and as in

```
SELECT PREVIOUS(R)
FROM ...
WHERE ...,
```

which retrieve the previous versions of elements.

- **Reconstruct(TEID)**: Reconstructs the tree rooted at EID in TEID for a particular version. The



timestamp TS in the TEID can be, for example, the result of NextTS/PreviousTS/CurrentTS operations. If previous versions of documents are stored as deltas, this can imply accessing and processing a potentially large number of deltas in addition to one complete version (the exception is the current version, which will normally be stored as a complete version). If previous versions of documents are stored as complete documents, only the actual version needs to be read.

- **Diff(E1, E2):** In some cases, querying for changes between different versions of elements is wanted. These changes can conveniently be returned (and eventually post-processed by the application or in a separate query) as edit scripts. Edit scripts describe changes between two versions, similar to, for example, the information in RCS files. In this context, the edit scripts are XML trees. Note that as long as an edit script is represented in XML, this operator does not break closure properties of queries. E1 and E2 can be versions of the same element but can also represent different documents or subtrees of elements. Diff is useful in constructions such as:

```
SELECT DIFF(R1,R2)
FROM ...
WHERE ....
```

It is possible to support string equality and string-contain queries with different operators and access structures. However, as described in Nørnvåg (2002), the access methods for containment queries already exist so that there is little to gain from providing additional access methods for string equality. Therefore, we expect that there are no separate operators and access structures for equality queries and that the general containment operators/access methods are used, followed by equality testing.

### Example Queries

The following three example queries based on the restaurant database example, together with the corresponding query operators, illustrate the use of the operators:

- **Q1:** List all restaurants in the list as of 26/01/2001:

```
SELECT R
FROM doc("http://guide.com/")/restaurant[26/01/2001] R
```

This is a snapshot query, listing the name in all versions of restaurant elements valid at time 26/01/2001 (i.e., versions created before or on 26/01/2001 that are not further updated or deleted).

- **Operators:** TPatternScan, followed by Reconstruct.
- **Q2:** Retrieve the number of restaurants at 26/01/2001:

```
SELECT SUM(R)
FROM doc("http://guide.com/")/restaurant[26/01/2001] R
```

- **Operators:** TPatternScan followed by the traditional aggregate operator Sum. Note that reconstruction of the documents is not needed. This is important, and shows that in many cases the storage of only deltas of previous document versions does not create performance problems.
- **Q3:** List the price history of the restaurant “Napoli”:

```
SELECT TIME(R), R/price
FROM doc("http://guide.com/")/restaurant[EVERY] R
WHERE R/name = “Napoli”
```

The use of EVERY instead of a particular timestamp retrieves all versions of restaurant. Note that the predicate in the WHERE clause acts on all versions, not only the current version of the elements. As a result, the price history of all restaurants through the history with the name Napoli will be listed.

- **Operator:** TPatternScanAll.

### Which Version?

An issue related to time is the assumed *content* at a particular time. Assume there is one document version from time T1, and another version of the document from T2 (where T2 > T1). What can we assume is the content at time T, where T1 < T < T2? Two possible options follow:

- Assume the contents for all T, where T1 < T ≤ T2 is the same as the contents at time T1.
- Assume that the document closest in time is the most appropriate (i.e., choose document version with timestamp Tx that minimizes |T - Tx|).

Frequently, it is possible to know the reasonable time approximation/version approximation to use because the information can be included in the query. This can be done

by extending the time expression that, as previously described, had the form [T] or [EVERY], where T is a timestamp:

- **[T MOST\_RECENT]:** Assume the contents for all T, where  $T_1 < T \leq T_2$  is the same as the contents of time  $T_1$ . This is expected to be the most common time approximation, and it is also similar to the previous approach. Therefore, it should be possible to denote by the short form [T].
- **[T NEAREST]:** Choose the document version with timestamp  $T_x$ , which minimizes  $|T - T_x|$ .

These language options can also be represented in the parameters to the operators.

## FUTURE TRENDS

The query operators described in this article, together with traditional query operators, are sufficient to execute temporal XML queries. However, several issues remain. First, efficient access methods are necessary. Second, whether ranking operations should also be included in the operator set should be studied further.

## CONCLUSION

This article has described how temporal queries can be executed in an XML database system. Issues related to time and identity, in this context, and described an appropriate data model as the basis of a temporal XML query language has been described. Temporal support in an XML query language implies that new operators are needed in query processing, and operators that will be useful to support typical queries in temporal XML databases have been identified. To achieve the desired performance in such databases, efficient execution of query operators are needed. For a more detailed discussion on these issues, see Nørnvåg (2002), which also describes algorithms for execution of the operators presented in this article.

## REFERENCES

Aguilera, V., Cluet, S., Veltri, P., Vodislav, D., & Watez, F. (2000). *Querying XML documents in Xyleme* (Tech. Rep. 182). Rocquencourt, France: Verso/INRIA.

Chawathe, S. S., Abiteboul, S., & Widom, J. (1998). Representing and querying changes in semistructured data. *Proceedings of the Fourteenth International Conference on Data Engineering* (pp. 4-13).

Chawathe, S. S., Abiteboul, S., & Widom, J. (1999). Managing historical semistructured data. *Theory and Practice of Object Systems*, 5(3), 143-162.

Grandi, F., & Mandreoli, F. (1999). *The valid Web: It's time to go* (Tech. Rep. TR-46). TimeCenter. Retrieved April 14, 2005, from <http://www.cs.aau.dk/TimeCenter/pub.htm>

Grandi, F., & Mandreoli, F. (2000). The valid Web: An XML/XSL infrastructure for temporal management of web documents. *Proceedings of Advances in Information Systems, First International Conference*, Izmir, Turkey.

Marian, A., Abiteboul, S., Cobena, G., & Mignet, L. (2001). Change-centric management of versions in an XML warehouse. *Proceedings of 27th International Conference on Very Large Data Bases* (pp. 581-590).

Nørnvåg, K. (2002). Algorithms for temporal query operators in XML. Paper presented at the *XML-Based Data Management and Multimedia Engineering EDBT 2002 Workshops, EDBT 2002 Workshops XMLDM, MDDE, and YRWS* (pp. 169-183).

Nørnvåg, K. (2003a). Space-efficient support for temporal text indexing in a document archive context. *Proceedings of the 7th European Conference on Digital Libraries*, Trondheim, Norway.

Nørnvåg, K. (2003b). V2: A database approach to temporal document management. *Proceedings of the 7th International Database Engineering and Applications Symposium*, (pp. 212-221).

Nørnvåg, K. (2004). Supporting temporal text-containment queries in temporal document databases. *Journal of Data & Knowledge Engineering*, 49(1), 105-125.

Nørnvåg, K., Limstrand, M., & Myklebust, L. (2003). TeXOR: Temporal XML database on an object-relational database system. *Proceedings of Perspectives of System Informatics*.

World Wide Web Consortium. (2001). XQuery: A query language for XML. Retrieved August 10, 2004, from <http://www.w3.org/TR/xquery/>

Xyleme, L. (2001). A dynamic warehouse for XML data of the Web. *IEEE Data Engineering Bulletin*, 24(2), 40-47.

## KEY TERMS

**Current Document Version:** The most recent version of a temporal document. Note that a deleted document has no current version.

**Full-Text Index:** An index supporting retrieval of identifiers from documents containing a particular word.

**Historical Document Version:** A non-current document version. This is versions that have later been updated, and the last version that existed before the document was deleted.

**Temporal Document:** A document in which the currently valid version and also the previously valid versions of the document are kept in the system. Every document version has an associated time period in which it was valid.

**Temporal Document Database Management System:** A system capable of storing, retrieving and querying

temporal documents. Typically, a document name or document identifier is used to distinguish documents from each other, and timestamp is used to distinguish particular versions of a document, as well.

**Temporal Full-Text Index:** An index supporting retrieval of identifiers from documents that contained a particular word at a particular point in time.

**Transaction-Time Temporal Document Database:** Every document version that is stored is assigned a timestamp (the commit time) by the system. The time period a particular version is valid, is from commit time until the document is deleted or updated by a new version.

**Valid-Time Temporal Document Database:** Every document that is stored is explicitly given a time period in which it is valid. The start and end timestamps of this period can be in the past, present, or future.

# Query Processing for RDF Data

**Heiner Stuckenschmidt**

*Vrije Universiteit Amsterdam, The Netherlands*

## INTRODUCTION

The World Wide Web today is a huge network of information resources which was built in order to broadcast information for human users. Consequently, most of the information on the Web is designed to be suitable for human consumption: the structuring principles are weak, many different kinds of information co-exist, and most of the information is represented as free text.

With the increasing size of the Web and the availability of new technologies, such as mobile applications or smart devices, there is a strong need for making the information on the World Wide Web accessible to computer programs which search, filter, convert, interpret, and summarize the information for the benefit of the user. The Semantic Web is a synonym for a World Wide Web whose accessibility is similar to a deductive database where programs can perform well-defined operations on well-defined data or even derive new information from existing data (Berners-Lee, Hendler & Lassila, 2001).

One of the main developments connected with the Semantic Web is the resource description framework RDF. RDF is an XML-based language for creating metadata about information resources on the Web. The metadata model is based on a resource which could be any piece of information with a unique name called URI (uniform resource identifier). URIs can either be unique resource locators (URLs)—well known from conventional Web pages—but also tagged information contained on a page or on other RDF definitions. The structure of RDF is very simple: a set of statement forms a labeled, directed graph where resources are represented by nodes and relations between resources by arcs. These are labeled with the name of the relation (Klyne & Carroll, 2004).

RDF, as such, only provides the user with a language for metadata. It does not make any commitment to a conceptual structure or a set of relations to be used. The RDF schema model defines a simple frame system structure by introducing standard relations like inheritance and instantiation, standard resources for classes, relations as well as a small set of restrictions on objects in a relation. Using these primitives, it is possible to define terminological knowledge about resources and

relations mentioned in an RDF model (Brickley & Guha, 2004).

## BACKGROUND

A characteristic property of an RDF model is that its statements form a labeled, directed graph. The resources mentioned in the statements can be seen as nodes in such a graph (this may include properties as special kinds of resources). Subject and object of each statement are connected by a directed link labeled with the name of the property acting as predicate.

Based on the graph-based view on an RDF model, we can characterize some properties of RDF models that are relevant for query answering. The first basic property of RDF is the fact that a graph entails all its subgraphs. The RDF data model described above and its associated semantics provides us with a basis for defining queries on RDF models in a straightforward way. The idea that has been proposed elsewhere and is adopted here is to use graphs with unlabeled nodes as queries. The unlabeled elements in a query graph can be seen as variables of the query. Answers to a query can be defined as:

- A subgraph of the given RDF model that is an instantiation of the query graph.
- The set of resources used to instantiate unlabeled nodes in the query graph.

From a theoretical point of view, these two definitions are exchangeable as one can easily be derived from the other by either extracting instantiated resources from the answer graph or by instantiating the query graph with the resources from the answer set, respectively.

One of the main features of RDF is the idea to enrich metadata descriptions with explicit models of their intended meaning. These models are defined in terms of a schema definition. We want to exploit this semantic information for query answering as it provides us with background information for computing more complete results. For this purpose, the RDF schema language contains properties with a standardized interpretation

Figure 1. An example RDF graph

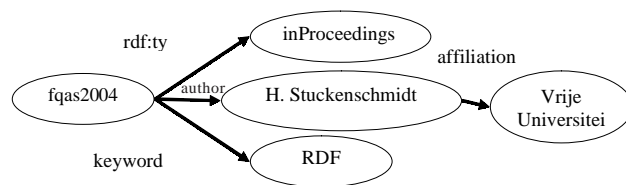
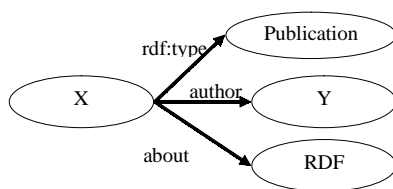


Figure 2. An example RDF query



for defining hierarchies of classes and properties as well as domain and range restrictions for user-defined properties. The intended meaning of these constructs and their impact on an RDF model is defined by the RDF semantics specification. The specification describes a model theoretic semantics and the notion of entailment for arbitrary RDF models (Hayes, 2004). It also defines a set of inference rules that can be used to derive implicit statements from a given RDF model. This implicit information should be taken into account when querying an RDF model making the query processing in the sense of a deductive database. A common approach is to compute the deductive closure of the model to be queried and execute the query on the resulting expanded model.

Figure 1 shows an RDF graph that represents a query. The nodes labeled X and Y are unlabeled nodes in the sense of the RDF model that acts as query variables. Assuming that *inProceedings* is a subclass of *Publication* and *keyword* is a subproperty of *about*, we see that the graph in Figure 1 represents a result of the query in Figure 2.

## RDF QUERY LANGUAGES AND SYSTEMS

A number of RDF query languages have been proposed that implement the general approach mentioned in the last section. Due to space limitations, we cannot discuss all of these languages. We focus on languages implemented in freely available RDF storage and retrieval systems.

## JENA and RDQL

The JENA system is a JAVA-based RDF infrastructure for parsing, storing, and accessing RDF data (<http://jena.sourceforge.com>). JENA is developed by the HP research labs in Bristol and implements the RDQL query language for accessing stored RDF data (<http://www.hpl.hp.com/semweb/rdql.htm>). An RDQL query consists of a SELECT part that specifies return variables, a WHERE part that refers to the RDF model to be accessed, a FROM part that provides a set of statement patterns that have to be matched by the RDF data in the model, a set of constraints on the variables occurring in the statement patterns, as well as a USING part where abbreviations for XML namespaces can be defined to simplify the query (Seabourne, 2004). In RDQL, schema awareness is not directly integrated in the language specification but has to be provided by the underlying data source.

## RDFSuite and RQL

The RDFSuite (Alexaki et al., 2002) is a toolkit for parsing, storing, and querying RDF data developed by ICS-FORTH in Greece (<http://139.91.183.30:9090/RDF/>). The RQL query language implemented in this suite is probably the most widely known and used RDF query language (<http://139.91.183.30:9090/RDF/RQL/>). RQL is based on functional query languages for object-oriented databases (in particular, OQL). Different from RDQL, the language provides a rich set of operators for specifying the query result that can freely be combined as a result of the functional nature of the language. These include explicit operators for navigating the schema (retrieving all super-classes or just the direct super-classes). Another distinguishing feature of RQL is the use of path expressions for navigating the RDF graph (Karvounarakis, Christophides & Plexousakis, 2001). This significantly reduces the number of equality constraints that are necessary in languages like RDQL for defining complex statement schemes.

## Sesame and SeRQL

The Sesame system (Broekstra, Kampman & van Harmelen, 2002) developed by the Dutch company Aduna is an RDF infrastructure that aims at providing RDF support on top of different information sources such as databases, search engines, and Web services (<http://www.openrdf.org/>). Besides providing support for the two above-mentioned languages, Sesame also implements its own query language SeRQL (<http://www.openrdf.org/publications/users/ch05.html>). The SeRQL language





combines features of RDQL and RQL aiming at ease of use rather than maximal expressiveness. A SeRQL query is structured similar to an RDQL query, but instead of a set of statement patterns, it uses path expressions that are similar to the ones used in RQL. SeRQL does not use rich operators as found in RQL and relies on the underlying information source to provide schema awareness. A special feature of SeRQL is the possibility to define the output format using a path expression which makes it possible to use SeRQL as a transformation language between different RDF model structures.

A common feature of most of the current systems is that they seek strong links with relational database technologies, both on the technical and conceptual level. All of the systems mainly use relational databases to store RDF data and try to push down complex queries into the database engine to make use of its optimization techniques. Normally, queries are translated into an equivalent SQL that is then executed on the underlying database. The translation varies significantly between the different systems, not only due to the differences in the RDF query languages but also due to different relational schemes used to store the RDF data.

## FUTURE TRENDS

While the work on defining the RDF data model and language is mainly finished by now, RDF query languages are still heavily discussed in the Semantic Web community. The same holds for RDF query processing. We can perceive some general lines of work that are in the focus of attention at the moment and will dominate upcoming research on RDF query processing.

### Standardization

In contrast to representation languages such as RDF and RDF schema, RDF querying currently suffers from a lack of agreement between the different approaches. Important query languages such as the ones described above are developed by individual groups and do not necessarily reflect the needs of a broader audience. As a result, the W3C has recently launched the RDF data access working group. The aim of the group is to create a standardized query language for RDF data and clarify its relationship to XML query languages.

### Optimization and Scalability

Compared to modern database systems, RDF storage technology still lacks sophisticated optimization methods for query processing. Current work in this direction is mainly

focussed on exploiting the optimization mechanisms of underlying database systems and on index structures for speeding up the access on triple level or for special queries. There are some attempts to fill this gap. Current work includes the definition of Algebras for RDF (Fransincar, Houben, Vdovjak & Barna), query planning, and result caching (Stuckenschmidt, 2004).

## Distributed Query Processing

Despite the inherently distributed nature of the Semantic Web, most current RDF infrastructures store information locally as a single knowledge repository; that is, RDF models from remote sources are replicated locally and merged into a single model. Distribution is virtually retained through the use of name spaces to distinguish between different models. Researchers only recently started to address the problem of querying distributed RDF data sources. Such a distributed setting requires additional mechanisms for data location, query planning, and result integration (Stuckenschmidt, Vdovjak, Broekstra & Houben, 2004).

## Ontology-Based Querying

Another recent trend is to investigate query processing with richer background knowledge than provided by the RDF schema language. Extensions of RDF schema with richer conceptual modelling features have been developed. The most important one is the Web Ontology Language OWL. The language consists of three increasingly expressive sublanguages. The weakest sublanguage, OWL lite, is similar to UML-like conceptual modelling languages. OWL DL, the second language is equivalent to expressive description logic (Borgida, 1995). The most expressive language combines description logic semantics with meta-modeling facilities. There are already results on querying databases with background knowledge (Peim, Franconi, Paton & Goble, 2002) and systems like JENA support limited support for OWL reasoning. Further, there are first proposals for a query language that allows the use of richer operators in queries (Fikes, Hayes & Horrocks, 2003).

## CONCLUSION

RDF technology is rapidly gaining importance as a standard for exchanging and accessing data in Web-based applications. In contrast to XML, RDF provides an application independent data model that eases the exchange of information between different systems on a syntactic

level. The ability to explicitly represent and access the underlying schema also helps to align information on a semantic level. These features, on the other hand, impose new challenges on the query processing for RDF data. Existing implementations already provide stable querying functionality and scale up to reasonably large models; RDF querying has not yet reached the level of maturity of traditional relational database systems. The standardization effort that is currently underway could be an important driver for work towards technologies that are comparable to traditional database technology.

## REFERENCES

- Alexaki, S., Athanasi, N., Christophides, V., Karvounarakis, G., Maganaraki, A., Plexousakis, D., & Tolle, K. (2002). The ICS-FORTH RDFSuite: High-level scalable tools for the Semantic Web. *Proceedings of the 11th International World Wide Web Conference (WWW2002)*, Honolulu, Hawaii, May 7-11.
- Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*, 5(1).
- Borgida, A. (1995, October). Description logics in data management. *IEEE Transactions on Knowledge and Data Engineering*, 7(5), 671-682.
- Brickley, D., & Guha, R.V. (2004). RDF vocabulary description language 1.0: RDF schema. Retrieved January 23, 2005, from <http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>
- Broekstra, J., Kampman, A., & van Harmelen, F. (2002). Sesame: A generic architecture for storing and querying RDF and RDF schema. *Proceedings of the Semantic Web (ISWC 2002)*, 2342 of LNCS (pp. 54-68).
- Fikes, R., Hayes, P., & Horrocks, I. (2003). *OWL-QL - A language for deductive query answering on the Semantic Web*. Knowledge Systems Laboratory, Stanford University, Stanford, CA.
- Frasincar, F., Houben, G.-J., Vdovjak, R., & Barna, P. (2004, March). RAL: An algebra for querying RDF. *World Wide Web*, 7(1), 83-109.
- Hayes, P. (2004). RDF semantics. Retrieved January 23, 2005, from <http://www.w3.org/TR/2004/REC-rdf-mt-20040210/>
- Karvounarakis, G., Christophides, V., & Plexousakis, D. (2001, December). RQL: A declarative query language for RDF. *D-Lib Magazine*, 7(12).
- Klyne, G., & Carroll, J.J. (2004). Resource description framework (RDF): Concepts and abstract syntax. Retrieved January 23, 2005, from <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>
- McGuinness, D., & van Harmelen, F. (2004). OWL Web ontology language overview. Retrieved January 23, 2005, from <http://www.w3.org/TR/2004/REC-owl-features-20040210/>
- Peim, M., Franconi, E., Paton, N., & Goble, C. (2002, July). Query processing with description logic ontologies over object-wrapped databases. *Proceedings of the 14th International Conference on Scientific and Statistical Database Management (SSDBM-02)*, Edinburgh, UK.
- Seaborne, A. (2004). RDQL - A query language for RDF. Retrieved January 23, 2005, from <http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/>
- Stuckenschmidt, H. (2004). Similarity-based query caching. In Christiansen et al. (Eds.), *Flexible query answering systems, 6th international conference (FQAS 2004)*, Lyon, France, June.
- Stuckenschmidt, H., Vdovjak, R., Broekstra, J., & Houben, G.-J. (2004). Data structures and algorithms for querying distributed RDF repositories. *Proceedings of the International World Wide Web Conference (WWW'04)*, New York, USA.

## KEY TERMS

**Blank Node:** A blank node is a node in the RDF graph that does not correspond to a Web resource. Blank nodes are normally used to refer to a set of resources or an entire statement. With respect to querying, they can be used to represent variable in an RDF query.

**OWL:** The Web Ontology Language, OWL, is a W3C recommendation for modeling complex conceptual models and background information in a Web-based setting. The language extends RDF schema with richer operators for defining classes of objects.

**RDF Graph:** An RDF model can be seen as a graph structure build-up by triples. The nodes of the graph are all resources and atomic values mentioned in the model. Nodes are linked by edges if they occur as subject and object in a triple contained in the corresponding RDF model.

**RDF Model Theory:** The RDF Model theory formally defines the interpretation of an RDF model using the notion of logical implication. The specification also defines a set of inference rules for computing implied statements.

**RDF Schema:** An RDF schema is an explicit representation of the conceptual model underlying an RDF model. The schema is represented in RDF using a set of properties with a standardized interpretation and effect on the interpretation of the model.

**Resource Description Framework (RDF):** The Resource Description Framework (RDF) is an XML based language for defining metadata for Web resources and relations between them. It is a W3C recommendation and serves as a basis for representing information and knowledge on the Semantic Web.

**Statement:** A statement is the basic structure found in an RDF specification. It consists of a subject, which is a Web resource represented by a unique identifier, and a predicate, which is a property that links the subject to an object which is either also a Web resource or an atomic value.

# Query Processing in Spatial Databases<sup>1</sup>

**Antonio Corral**

*University of Almeria, Spain*

**Michael Vassilakopoulos**

*TEI of Thessaloniki, Greece*

## INTRODUCTION

Spatial data management has been an active area of intensive research for more than two decades. In order to support spatial objects in a database system, several important issues must be taken into account such as spatial data models, indexing mechanisms, and efficient query processing. A spatial database system (SDBS) is a database system that offers spatial data types in its data model and query language and supports spatial data types in its implementation, providing at least spatial indexing and efficient spatial query processing (Güting, 1994).

The main reason that has caused the active study of spatial database management systems (SDBMS) comes from the needs of the existing applications, such as geographical information systems (GIS), computer-aided design (CAD), very large-scale integration design (VLSI), multimedia information systems (MIS), data warehousing, and so forth.

Some of the most important companies in the commercial database industry (Oracle, Informix, Autodesk, etc.) have products specifically designed to manage spatial data. Moreover, research prototypes as Postgres and Paradise offer the possibility to handle spatial data. The main functionality provided by these products includes a set of spatial data types such as the point, line, polygon, and region, and a set of spatial operations, including intersection, enclosure, and distance. The performance enhancement provided by these operations includes spatial access methods and query algorithms over such indexes (e.g., spatial range queries, spatial joins, etc.). We must also cite the Open Geographic Information Systems (OGIS) consortium (<http://www.opengis.org/>), which has developed a standard set of spatial data types and operations and SQL3/SQL99, which is an object-relational query language that provides the use of spatial types and operations.

In a spatial database system, the queries are usually expressed in a high-level declarative language such as SQL; therefore, specialized database software has to map the query in a sequence of spatial operations supported by spatial access methods (Shekhar & Chawla, 2003).

Spatial query processing refers to the sequence of steps that a SDBMS will initiate to execute a given spatial query. The main target of query processing in the database field is to process the query accurately and quickly (consuming the minimum amount of time and resources on the computer), by using both efficient representations and efficient search algorithms (Graefe, 1993). Query processing in a spatial environment focuses on the design of efficient algorithms for spatial operators (e.g., selection operations, spatial joins, distance-based queries, etc.). These algorithms are both CPU and I/O intensive, despite common assumptions of traditional databases that the I/O cost will dominate CPU cost (except expensive distance-based queries), and therefore an efficient algorithm is one that minimizes the number of disk accesses.

## BACKGROUND IN SPATIAL QUERIES AND PROCESSING

From the query processing point of view, the following three properties characterize the differences between spatial and relational databases (Brinkhoff, Kriegel & Seeger, 1993): (1) unlike relational databases (Elmasri & Navathe, 2000), spatial databases do not have a fixed set of operators that serve as building blocks for query evaluation; (2) spatial databases deal with extremely large volumes of complex objects, which have spatial extensions and cannot be sorted in a one-dimensional array; (3) computationally expensive algorithms are required to test the spatial operators, and the assumption that I/O costs dominate CPU costs is no longer valid.

We generally assume that the given spatial objects are embedded in  $d$ -dimensional Euclidean space ( $E^d$ ). An object  $obj$  in a spatial database is usually defined by several non-spatial attributes and one attribute of some spatial data type (point, line, polygon, region, etc.). This spatial attribute describes the geometry of the object  $obj.G \subseteq E^d$ , that is, the location, shape, orientation, and size of the object. The most representative spatial operations, which are the basis for the query processing in spatial databases, are (1) update operations; (2) selection operations (point and range queries); (3) spatial join; and

(4) spatial aggregate queries (Gaede & Günther, 1998; Shekhar & Chawla, 2003).

- **Update Operations:** Standard database operations such as modify, create, and so forth.
- **Point Query (PQ):** Given a query point  $p \in E^d$ , find all spatial objects  $O$  that contain it.
- **Range Query (RQ):** Given a query polygon  $P$ , find all spatial objects  $O$  that intersect  $P$ . When the query polygon is a rectangle, this is called a window query.
- **Spatial Join Query (SJQ):** Given two collections  $R$  and  $S$  of spatial objects and a spatial predicate  $\theta$ , find all pairs of objects  $(O, O') \in R \times S$  ( $O \in R$  and  $O' \in S$ ), where  $\theta(O, O')$  evaluates to true. Some examples of the spatial predicate  $\theta$  are intersects, contains, is\_enclosed\_by, distance, northwest, adjacent, meets, and so on. For spatial predicates such as contains, encloses, or adjacent, for example, the intersection join is an efficient filter that yields a set of candidate solutions typically much smaller than the Cartesian product  $R \times S$ . An extension of the intersection join is the multiway spatial join, which involves an arbitrary number of spatial inputs (Mamoulis & Papadias, 2001). Very interesting distance join queries are actually being studied, for example, closest pairs query (Corral et al., 2000), buffer query (Chan, 2003), nearest neighbors join (Böhm & Krebs, 2002), iceberg queries (Shou et al., 2003), distance join queries of multiple inputs (Corral et al., 2003), and so on.
- **Spatial Aggregate Queries (SAQ):** This kind of spatial query involves specifying a region of space and asking for the value of some aggregate function (e.g., count, sum, min, max, average) for which we have measurements for this given region (Papadias et al., 2001). Spatial aggregates are usually variants of the nearest neighbor problem (Shekhar & Chawla, 2003). The Nearest Neighbor Query (NNQ) has the form: given a spatial object  $O'$ , find all spatial objects  $O$  having a minimum distance from  $O'$ . The distance between extended spatial data objects is usually defined as the distance between their closest points (common distance functions for points include the Euclidean and the Manhattan distance). An interesting variant of NNQ is the reverse nearest neighbor query (RNNQ), which reports the points that have the query point as their nearest neighbor (Korn & Muthukrishnan, 2000).

The spatial queries are often processed using filter and refine techniques to minimize both the CPU and I/O cost (Brinkhoff et al., 1994). Approximate geometry such as the minimal orthogonal bounding rectangle (MBR) of an extended spatial object is first used to filter out many

irrelevant objects quickly. An MBR is characterized by *min* and *max* points of hyper-rectangles with faces parallel to the coordinate axes. Using the MBR instead of the exact geometrical representation of the spatial object, its representational complexity is reduced to two points, where the most important object features (position and extension) are maintained. The R-tree (Guttman, 1984) is a spatial access method that represents the spatial objects by their MBR, and it is a height-balanced tree. Therefore, in the filter step, many candidates are eliminated using the spatial predicate and the MBRs of the spatial objects. In the refinement step, the exact geometry of each spatial object from the candidate set (result of the filter step) and the exact spatial predicate are examined. This step usually requires the use of CPU-intensive algorithms like computational geometry algorithms for spatial operations (Rigaux, Scholl & Voisard, 2001). Strategies for range-queries include a scan and index-search in conjunction with the plane-sweep algorithm (Brinkhoff et al., 1993). Strategies for the spatial join include the nested loop, tree matching (Brinkhoff et al., 1993; Huang, Jing & Rundensteiner, 1997), when indices are present on all participating inputs and space partitioning (Arge et al., 1998; Lo & Ravishankar, 1996; Patel & DeWitt, 1996) in absence of indexes. For the case when one spatial input is indexed, the most representative join strategies have been proposed by Lo & Ravishankar (1994) and Mamoulis & Papadias (2003).

Nearest neighbor queries (NNQ) are common in many applications, for example, GIS, pattern recognition, document retrieval, and learning theory. As the previous spatial queries, NNQ algorithms are also two-step algorithms (filter-refine paradigm), and they follow branch-and-bound techniques, using distance functions and pruning heuristics in order to reduce the search space. The most representative algorithms to perform NNQ over spatial data have been proposed by Roussopoulos, Kelley, and Vincent (1995) and Hjaltason and Samet (1999) on R-trees. The first query algorithm follows a depth-first traversal, whereas the second one is an incremental algorithm following a best-first search on the R-tree. These algorithms can be extended to find  $K$ -nearest neighbors by slight modification of the pruning rules to retain the  $K$  best candidates.

## PERSPECTIVE AND NEW IMPORTANT ISSUES

We have reviewed the most representative spatial queries, using mainly the overlap predicate for range queries and spatial join queries. However, there is a need to develop and evaluate query strategies for many other frequent spatial queries that can be demanded by the





Table 1. A list of new spatial queries

<i>Buffer</i>	Find the areas 500 meters from power lines
<i>Voronoi</i>	Classify households as to which supermarket they are closest to
<i>Neighborhood</i>	Determine slope based on elevation
<i>Network</i>	Find the shortest path from the warehouse to all delivery stops
<i>Allocation</i>	Where is the best place to build a new restaurant?
<i>Transformation</i>	Triangulate a layer based on elevation
<i>Ranking</i>	Find the top-k hotels with the largest number of nearby restaurants
<i>Chromatic</i>	Find the type of monument nearest to the Eiffel Tower
<i>Aggregate</i>	Find the total number of (restaurant, hotel) pairs that are within 1 km from each other
<i>Multi-way</i>	Find all cities within 100 km of Madrid crossed by a river which intersects an industrial area

users in a spatial database system. Table 1 summarizes some of these new spatial queries, which include queries on objects using predicates other than overlap and queries on fields such as slope analysis as well as queries on networks, and so forth (Shekhar et al., 1999).

The ever-increasing demand and easy availability of the Internet have prompted the development of Web-based Geographic Information Systems (WGIS) for easy sharing of spatial data and executing spatial queries over the Internet using Web environments like Web servers, Web clients, common data formats (HTML, XML), common communication protocols (http), uniform resource locator (URL), and so forth. In a WGIS architecture (Shekhar & Chawla, 2003), the GeoSpatial Database Access Layer (GSDAL) allows access to the spatial data using a SDBMS, where efficient query processing is required. In order to improve Web-Based Spatial Database Systems (WSDBS), new important issues for SDBMSs and Web technology need to be addressed. For example, to offer SDBMS services on the Web, evaluate and improve the Web for SDBMS clients and servers, use Web data formats for spatial data (e.g., VMS and GML), employ safe communication protocols, allocate adequate search facilities on the Web, develop compatibility between several WSDBS, improve maintenance and integrity of data, and so on.

## FUTURE TRENDS

Most of the spatial query algorithms have been studied over point data sets in the 2-dimensional space, and a natural extension is to study their application on data sets containing objects with spatial extent (lines, polygons, regions, 3D objects, etc.). In order to carry out this work, geometric algorithms (based on computational geometry) have to be designed and analyzed for spatial operators using these complex spatial objects (Rigaux et al., 2001).

Many open research areas exist at the logical level of query processing, including query-cost modeling and

queries related to spatial networks. Cost models are used to rank and select the promising processing strategies, given a spatial query and a spatial data set. Traditional cost models may not be accurate in estimating the cost of strategies for spatial operations, mainly due to the distance metric. Cost models are needed to estimate the selectivity of spatial search and join operations toward comparison of execution-costs of alternative processing strategies for spatial operations during query optimization. Preliminary work in the context of the R-tree, using fractal-models for NNQ (Belussi & Faloutsos, 1998), using the concept of Minkowski sum for RQ and NNQ (Böhm, 2000), and using the concept of density of a rectangle set for RQ and SJQ (Theodoridis, Stefanakis & Sellis, 2000) have been developed, but more work is needed.

Spatial network databases are an important component of SDBS, since this is the kernel of many important real-life applications as transportation planning, air traffic control, urban management, electric, telephone and gas networks, river transportation, and so forth. Previous efforts in this research area have been focused on disk-based graph representation as the connectivity clustered access method (CCAM) (Shekhar & Chawla, 2003), nearest neighbor queries in road networks by transforming the problem to a high dimensional space (Shahabi, Kolahdouzan & Sharifzadeh, 2002), and a flexible architecture that integrates network representation (preserving connectivity and locations) and Euclidean restriction for processing the most common spatial queries (range search, nearest neighbors, spatial joins, and closest pairs) (Papadias et al., 2003). Interesting research has been developed in this field, but much more work is needed mainly in measuring the I/O cost for network operations in new queries (e.g., *find the two nearest gas stations, in terms of network distance, along the route from Madrid to Barcelona*), or in moving objects environment (e.g., *find the closest taxi to our present location*), and so on.

## CONCLUSION

Spatial query processing refers to the sequence of steps that a SDBMS will initiate to execute a given spatial query. The main target of query processing in the database field is to process the query accurately and quickly by using both efficient representations and efficient search algorithms. Query processing in a spatial environment focuses on the design of efficient algorithms for spatial operators (e.g., selection operations, spatial joins, etc.). Spatial query operations can be classified into four groups: *point*, *range*, *spatial join*, and *spatial aggregate*. For spatial query processing, the filter-refine paradigm is used over spatial access methods to minimize both the CPU and I/O cost. Future research trends include the study of new spatial queries (especially on spatial networks), the study of issues related to Web-Based Spatial Database Systems, and work on cost models for estimating the selectivity of spatial queries.

## REFERENCES

- Arge, L., Procopiuc, O., Ramaswamy, S., Suel, T., & Vitter, J.S. (1998). Scalable sweeping-based spatial join. *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB 1998)*, New York, August 24-27.
- Belussi, A., & Faloutsos, C. (1998). Self-spacial join selectivity estimation using fractal concepts. *ACM Transactions on Information Systems*, 16(2), 161-201.
- Böhm, C. (2000). A cost model for query processing in high dimensional data spaces. *ACM Transactions on Database Systems*, 25(2), 129-178.
- Böhm, C., & Krebs, F. (2002). High performance data mining using the nearest neighbor join. *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002)*, Maebashi City, Japan, December 9-12.
- Brinkhoff, T., Kriegel, H.P., Schneider, R., & Seeger, B. (1994). Multi-step processing of spatial joins. *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data (SIGMOD 1994)*, Minneapolis, Minnesota, May 24-27.
- Brinkhoff, T., Kriegel, H.P., & Seeger, B. (1993). Efficient processing of spatial joins using R-Trees. *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD 1993)*, Washington, DC, May 26-28.
- Chan, E.P.F. (2003). Buffer queries. *IEEE Transactions on Knowledge Data Engineering*, 15(4), 895-910.
- Corral, A., Manolopoulos, Y., Theodoridis, Y., & Vassilakopoulos, M. (2000). Closest pair queries in spatial databases. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD 2000)*, Dallas, Texas, May 16-18.
- Corral, A., Manolopoulos, Y., Theodoridis, Y., & Vassilakopoulos, M. (2003). Distance join queries of multiple inputs in spatial databases. *Proceedings of Advances in Databases and Information Systems (ADBIS 2003)*, Dresden, Germany, September 3-6.
- Elmasri, R., & Navathe, S. (2000). *Fundamentals of database systems*. Addison-Wesley/Benjamin Cummings.
- Gaede, V., & Günther, O. (1998). Multidimensional access methods. *ACM Computing Surveys*, 30(2), 170-231.
- Graefe, G. (1993). Query evaluation techniques for large databases. *ACM Computing Surveys*, 25(2), 73-170.
- Güting, R. (1994). An introduction to spatial database systems. *VLDB Journal*, 3(4), 357-399.
- Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data (SIGMOD 1984)*, Boston, June 18-21.
- Hjaltason, G.R., & Samet, H. (1999). Distance browsing in spatial databases. *ACM Transactions on Database Systems*, 24(2), 265-318.
- Huang, Y.W., Jing, N., & Rundensteiner, E.A. (1997). Spatial joins using R-trees: Breadth-first traversal with global optimizations. *Proceedings of 23rd International Conference on Very Large Data Bases (VLDB 1997)*, Athens, Greece, August 25-29.
- Korn, F., & Muthukrishnan, S. (2000). Influence sets based on reverse nearest neighbor queries. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD 2000)*, Dallas, Texas, May 16-18.
- Lo, M.L., & Ravishankar, C.V. (1994). Spatial joins using seeded trees. *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data (SIGMOD 1994)*, Minneapolis, Minnesota, May 24-27.
- Lo, M.L., & Ravishankar, C.V. (1996). *Spatial hash-joins*. *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data (SIGMOD 1996)*, Montreal, Quebec, Canada, June 4-6.
- Mamoulis, N., & Papadias, D. (2001). Multiway spatial joins. *ACM Transactions on Database Systems*, 26(4), 424-475.

Mamoulis, N., & Papadias, D. (2003). Slot index spatial join. *IEEE Transactions on Knowledge Data Engineering*, 15(1), 211-231.

Papadias, D., Kalnis, P., Zhang, J., & Tao, Y. (2001). Efficient OLAP operations in spatial data warehouses. *Proceedings of the Symposium on Spatial and Temporal Databases (SSTD 2001)*, Redondo Beach, California, July 12-17.

Papadias, D., Zhang, J., Mamoulis, N., & Tao, Y. (2003). Query processing in spatial network databases. *Proceedings of the Very Large Data Bases Conference (VLDB 2003)*, Berlin, Germany, September 9-12.

Patel, J.M., & DeWitt, D.J. (1996). Partition based spatial-merge join. *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data (SIGMOD 1996)*, Montreal, Quebec, Canada, June 4-6.

Rigaux, P., Scholl, M.O., & Voisard, A. (2001). *Spatial databases: With application to GIS*. Morgan Kaufmann.

Roussopoulos, N., Kelley, S., & Vincent, F. (1995). Nearest neighbor queries. *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data (SIGMOD 1995)*, San Jose, California, May 22-25.

Shahabi, C., Kolahdouzan, M., & Sharifzadeh, M. (2002). A road network embedding technique for k-nearest neighbor search in moving object databases. *Proceedings of the 1996 ACM Symposium on Advances in Geographic Information Systems (ACM GIS 1996)*, McLean, Virginia, November 8-9.

Shekhar, S., & Chawla, S. (2003). *Spatial databases: A tour*. Prentice Hall.

Shekhar, S., Chawla, S., Ravada, S., Fetterer, A., Liu, X., & Lu, C.T. (1999). Spatial databases: Accomplishments and research needs. *IEEE Transactions on Knowledge Data Engineering*, 11(1), 45-55.

Shou, Y., Mamoulis, N., Cao, H., Papadias, D., & Cheung, D.W. (2003). Evaluation of iceberg distance joins. *Proceedings of the 8th International Symposium on Spatial and Temporal Databases (SSTD 2003)*, Santorini Island, Greece, July 24-27.

Theodoridis, Y., Stefanakis, E., & Sellis, T. (2000). Efficient cost models for spatial queries using R-trees. *IEEE Transactions on Knowledge Data Engineering*, 12(1), 19-32.

## KEY TERMS

**Buffer Query:** This spatial query involves two spatial datasets and a distance threshold  $\Delta$ . The answer is a set

of pairs of spatial objects from the two input datasets that are within distance  $\Delta$  from each other.

**Filter-Refine Paradigm:** Algorithms that follow this paradigm are two-step algorithms. *Filter step:* an approximation of each spatial object is used to produce a set of candidates (and, possibly, a set of actual answers), which is a superset of the answer set consisting of actual answers and false hits. *Refinement step:* each candidate from the filter step is then examined with respect to its exact geometry in order to produce the answer set by eliminating false hits.

**Iceberg Distance Join:** This spatial query involves two spatial datasets, a distance threshold  $d$ , and a cardinality threshold  $K$  ( $K \geq 1$ ). The answer is a set of pairs of objects from the two input datasets that are within distance  $\Delta$  from each other, provided that the first object appears at least  $K$  times in the join result.

**K Closest Pairs Query:** This spatial query involves two spatial datasets and a cardinality threshold  $K$  ( $K \geq 1$ ). It discovers the  $K$  distinct pairs of objects from the two input datasets that have the  $K$  smallest distances between them.

**K Nearest Neighbors Join:** This spatial query involves two spatial datasets and a cardinality threshold  $K$  ( $K \geq 1$ ). The answer is a set of pairs from the two input datasets that includes, for each of the spatial objects of the first dataset, the pairs formed with each of its  $K$  nearest neighbors in the second dataset.

**Spatial Data Types:** Spatial data types provide a fundamental abstraction for modeling the structure of geometric entities in space (geometry) as well as their relationships (topology), for example, points, lines, polygons, regions, and so forth. A *spatial object* is an object with at least one attribute of a spatial data type.

**Spatial Database System (SDBS):** A spatial database system is a database system that offers spatial data types in its data model and query language and supports spatial data types in its implementation, providing at least spatial indexing and efficient spatial query processing.

**Spatial Operators:** Spatial operators represent the spatial relationships between spatial objects. The most representative spatial relationships are: (1) topological relationships, such as adjacent, inside, disjoint, and so forth are invariant under topological transformations like translation, scaling, and rotation; (2) direction relationships, for example, above, below, north\_of, southwest\_of, and so forth; and (3) metric relationships, for example, distance  $< 100$ .

**Spatial Query:** It is a set of spatial conditions characterized by spatial operators that form the basis for the

retrieval of spatial information from a spatial database system.

**Spatial Query Processing:** It focuses on extracting information from a large amount of spatial data without actually changing the spatial database. It is different to the concept of query optimization that focuses in finding the best query evaluation plan that minimizes the most relevant performance measure (e.g. CPU, I/O, etc.).

## ENDNOTE

- <sup>1</sup> Supported by the ARCHIMEDES project 2.2.14, «Management of Moving Objects and the WWW» of the Technological Educational Institute of Thessaloniki (EPEAEK II), co-funded by the Greek Ministry of Education and Religious Affairs and the European Union.

# Raster Databases

Peter Baumann

International University Bremen, Germany

## INTRODUCTION

Spatio-temporal data play an important role in science, both as observed natural phenomena like temperature curves or satellite imagery and as artificially generated data such as simulation results or statistical data derived from spatio-temporal phenomena. As a coarse but common classification, spatio-temporal data can be grouped into discretized and conceptually continuous data (Figure 1). The first category allows points, lines, areas, and bodies to have any coordinate value, while the latter category has all data values sitting at the crosspoints of equidistant grids. When dealing with maps, these categories are called vector and raster data, respectively, while in Computational Fluid Dynamics (CFD), for example, the terms general mesh and regular mesh are in use.

We will use the term *raster/sampled/discretized data* or *Multidimensional Discrete Data* (MDD) interchangeably with the definition that such an object consists of a set of point/value pairs where the points fill an axis-parallel rectangular area in the Euclidean space  $\mathbf{Z}^d$  for some dimension  $d \geq 1$ . Obviously, this structure is equivalent to an array in the programming language sense.

MDD appear in a large variety of applications. Examples for multidimensional raster data are 1-D scalar measurements like temperature and radioactivity, 2-D satellite imagery spanning large seamless maps of the Earth's surface, 3-D image time series ( $x/y/t$ ) and geophysical data ( $x/y/z$ ), and 4-D climate models ( $x/y/z/t$ ). Figure 2 shows some of today's most important areas; each field in turn usually has many diverse subfields, as the example of mapping/cartography shows.

The kind of services required on MDD can be summarized as *fast, flexible selection on huge raster data assets*.

Figure 1. Discretized raster vs. continuous spatial objects

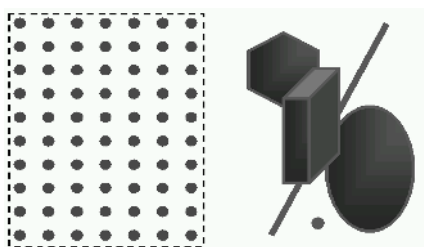
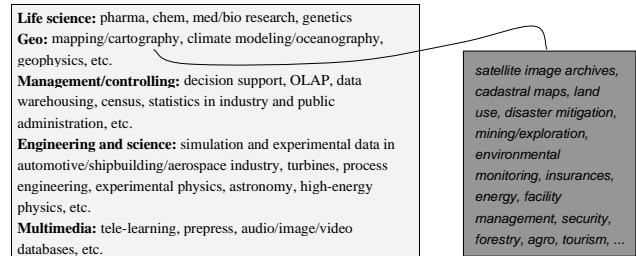


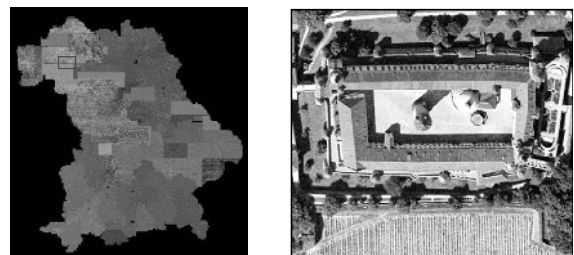
Figure 2. Raster application fields



For example, in Figure 3, the aerial image of Bavaria is depicted, consisting of 950,000 x 1,000,000 RGB (red/green/blue) pixels; this represents a raw data volume of 2.85 Terabyte (TB). A common task on such geo-imagery is to interactively select and display the overall image or a small cutout scaled to the client's window size (see Figure 3), something which occupies, say, 30 kB as a JPEG image. Further operations involve analysis like deriving the vegetation index from multi-band satellite images.

Recent hardware development allows holding of such large objects available for online access, and there is an increasing demand for Web services including MDD support, such as aerial/satellite image archives (see [www.ceos.org](http://www.ceos.org), for example). Consequently, database systems (DBMSs) in future will have to provide storage, query, and optimization support to accommodate MDDs side by side with the traditional data types — in database speak, MDDs must become first-class citizens in the database. In the sequel, we discuss how this can be accomplished and what issues are still waiting to be solved.

Figure 3. Aerial image of Bavaria: downscaled (thumbnail) overall view (left) and zoomed cutout (right)





## BACKGROUND

### Differentiation to Other Fields

A related but different field where databases also handle images is Multimedia Databases. Multimedia database systems rely on content recognition techniques to extract semantic knowledge from the image beforehand and henceforth perform all querying on this semantic net leaving the imagery untouched except for displaying them unchanged. Raster databases, conversely, work on the pixel level and do not attempt to understand the contents; rather, they allow quick navigation on and selection from very large data objects.

Another related field is image processing. While the set of operations known there exceeds raster database functionality by far, data sets in imaging systems traditionally have to fit into main memory; raster databases, on the other hand, focus on data selection on objects which may well exceed main memory capacity by a factor of a thousand or more.

### History and State of the Art

Work on raster databases—whether industrial or scientific—falls into two basic categories: statistics and sensor/image databases. The field of statistical databases has received outstanding thrust through data warehousing and online analytical processing (OLAP) which use model business data as cell values (“facts”) allocated in multidimensional spaces (“data cubes”) described by abstract dimensions (“features”).

Completely separate from this, sensor and image data have been investigated. Traditionally, images have been stored in *BLOBs* (binary large objects), that is, byte strings without any further semantics, introduced as “long fields” by Lorie (1982). First approaches to add more semantics include Tamura (1980) where a set of imaging functions was added to the programming interface, but not to the query language; there was no conceptual background justifying the operations chosen. A first image query language was proposed with PICDMS (Chock, Cardenas & Klinger, 1984; Joseph & Cardenas, 1988). However, many queries were dependent on the operation sequence, and no architectural support for large objects was indicated. In Vandenberg and DeWitt (1991), a general-purpose conceptual database model was extended with simple array capabilities. The quest for support of non-trivial array operations in a query language was first phrased in Buneman (1993). A conceptual raster model with a declarative, optimizable query language based on an algebraic framework was presented in Baumann (1994, 1999); this approach has been implemented in the *rasdaman*

system ([www.rasdaman.com](http://www.rasdaman.com)) which is in worldwide commercial use. Other algebras which have been implemented to a lesser extent are Libkin, Machlin, and Wong (1996) and Marathe and Salem (1997, 1999).

An example for domain-specific DBMS extensions to accommodate, in this case, 3D medical imagery is described in Arya et al. (1994). Requirements for supercomputing data management have been stated in Kleese (2000) from an application point of view.

With their version 10g, Oracle ([www.oracle.com](http://www.oracle.com)) has released raster support for large 2-D geographic imagery.

The main difference between statistical and sensor/image databases does not lie in the data structure (both deal with multidimensional grids), nor are operations substantially different (an OLAP roll-up from days to weeks is mathematically close to scaling an image by a factor of 7). The essential difference lies in the *sparsity*, that is, the percentage of cells in the data space considered which actually carry a value. Statistical databases are *sparsely populated* (on the average about 2%, maximum 5%), while image data usually are *densely populated* (usually between 60% and 100%). Technology to handle such data is completely different for both cases. However, given the far-going similarity between both, it seems promising to research ways for an integrated approach.

### Relevant Bodies

- *SQL/MM* defines database handling of imagery in the context of the *SQL* standard.
- The *OpenGIS Consortium* (OGC, [www.opengis.org](http://www.opengis.org)) standardizes interfaces for Web-based services on geographic information, among them, multidimensional raster (“coverage”) data in the *Web Coverage Service* (WCS) standard.
- *CODATA* (Committee on Data for Science and Technology, [www.codata.org](http://www.codata.org)) is a user-driven international organization whose goal is to enhance accessibility of scientific data.
- *ERCOFTAC* (European Research Community On Flow, Turbulence And Combustion, [www.ercoftac.org](http://www.ercoftac.org)) coordinates data management research and provides sample data sets in the field of Computational Fluid Dynamics (CFD).

## A SAMPLE RASTER SERVER

As an example for multidimensional raster database support, we sketch the *rasdaman* system which has been implemented in the course of several European-funded research projects and has been commercialized mainly in the field of geographic image map services. The underlying

ing formal basis, rasdaman array algebra, has been influenced by Image Algebra (Ritter, Wilson & Davidson, 1990).

The conceptual model of rasdaman centers around the notion of typed n-D arrays augmented with an *object identifier* (OID). Such MDD objects are maintained in collections (sets) similar to relational tables. Arrays are defined through a template `marray<b,d>` which is instantiated with the array base type `b` and the array extent (*spatial domain*) `d`, specified by the lower and upper bound for each dimension. Thus, a 2-D color image of unbounded domain can be defined as:

```
typedef marray
< struct{ char red, green, blue; },
  [ ** , ** ]
> MyImage;
```

### Raster Retrieval

The rasdaman query language, `rasql`, extends SQL with multidimensional imaging expressions. Like SQL, a `rasql` query always returns a set of items (in this case, MDD objects). The expressive power has been limited to non-recursive operations, thereby guaranteeing termination of any well-formed query. Below we provide a brief overview on `rasql`.

- **Subsetting:** This includes *trimming* (rectangular cutouts) and *section* (extraction of lower-dimensional sub-arrays).

**Example 1:** The following query retrieves a 2000x3000 cutout from every Landsat image in `LandsatCollection`:

```
SELECT c[ 1001:3000, 1001:4000 ]
FROM LandsatCollection AS c
```

**Example 2:** Let us assume that `ClimateCollection` contains one 4-D cube. The following query, then, extracts a 3-D volume at time frame 1020 from this cube:

```
SELECT c[ 1020, ** , ** , ** ]
FROM ClimateCollection AS c
```

- **Induced Operations:** For each operation available on the MDD cell type, a corresponding so-called induced operation is provided which simultaneously applies the base operation to all cells of an MDD. Both unary (e.g., record access or contrast enhancement) and binary operations (e.g., masking an image) can be induced.

**Example 3:** Figure 4 shows the three visible bands of a satellite view on the Spanish coast. The query below masks out all pixels from this image which are considered non-vegetation due to their color value.

```
SELECT c * (c.green > 130 AND c.red < 110 AND
c.blue < 140)
FROM LandsatCollection AS c
```

In general, MDD expressions can be used in the `SELECT` part of a query and, if the outermost expression result type is Boolean, also in the `WHERE` part.

- **Deriving Summary Data:** Condenser operations—in standard SQL called aggregation—allow summarization over all values from some spatial domain. The most common statistical operators are provided as shorthands; others can be formulated as MDD expressions.

**Example 4:** “Average green intensity within a Landsat scene.”

```
SELECT avg_cells(c.green)
FROM LandsatCollection AS c
```

### Array Storage

The storage management’s task is to efficiently map the conceptual entities (collections of MDD objects) to some appropriate storage structure. Basically, three alternatives for representing an MDD object are known today:

- **Sequential storage of cell values following some linearization scheme (e.g., row-major):** Coordinates need not be stored since cell locations can be computed following the bijective linearization scheme. This is very efficient for dense MDD; for sparse data, it is less advantageous because many non-existing values have to be materialized, thereby

Figure 4. Landsat scene (left) and vegetation highlighted (right)

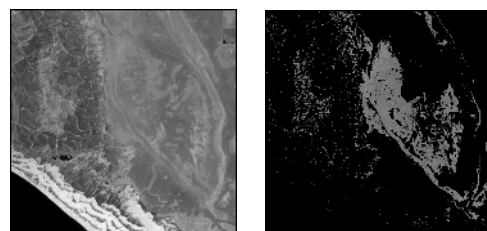


Figure 5. Row-major linearization of a 2-D cell array, followed by compression; “X” cells show some region getting scattered on disk, thereby rendering all but horizontal access inefficient

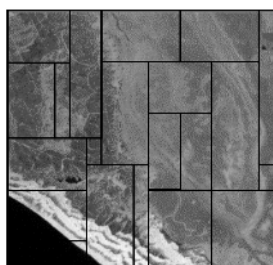


significantly increasing storage need. Further, the core advantage, addressability without materializing coordinates, is lost at the moment compression is applied. Finally, depending on the linearization scheme chosen, access in a particular direction is favored (e.g., horizontal reading) while all other access patterns are severely degraded.

- **Sets of tuples containing (coordinate, value) pairs:** Obviously, this technique, which is used in relational OLAP (ROLAP), is particularly suitable for sparse data—the overhead of storing coordinates is more than compensated by avoiding materialization of non-existing values. Conversely, no one would seriously consider storing (dense) images this way.
- **Hybrid, two-level approaches:** There, MDD objects are partitioned into sub-arrays called tiles or chunks. Tiles are stored together with their coordinate extent; inside, each tile value is stored linearized. Tiles this way naturally form the unit of storage access. Cell access is fast, except when tile boundaries are crossed and another tile has to be loaded. Note that the first two alternatives form special cases of this method. In the case that the partition consists of only one item, we obtain the linearized scheme; if each partition contains exactly one cell, we have the set-of-tuples case. Actually, this partitioning method is not new at all; it has been used in the imaging community for decades to handle images larger than main memory.

In rasdaman, raster objects are maintained in a standard relational database following the tiling approach (Furtado, 1999). Aside from regular grids, any user or

Figure 6. Sample 2-D tiling



system generated partitioning is possible (Figure 6). A *geo index* is employed to quickly determine the tiles affected by a query. Optionally, tiles are compressed for storage using one of various choices; additionally, query results can be compressed for transfer to the client. Both tiling strategy and compression comprise database tuning parameters. Tiles and index are stored as BLOBs in a relational database which also holds the data dictionary needed by rasdaman’s dynamic type system. Hence, the RDBMS simply acts as a persistent store for moderately sized BLOBs with OIDs, allowing use of virtually any DBMS.

### Query Evaluation

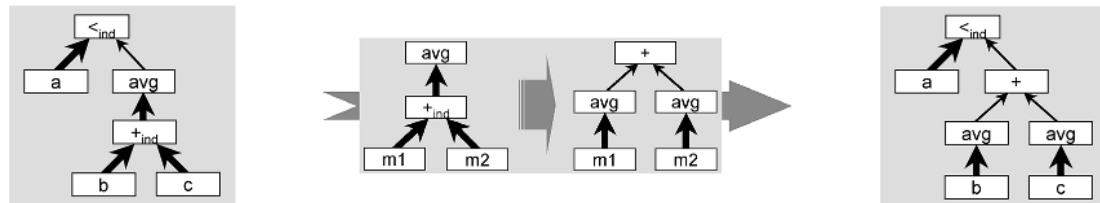
Queries are parsed, optimized, and executed in the rasdaman server. The parser receives the query string and generates the operation tree. Query evaluation is tile based meaning that, whenever possible, each MDD object affected is inspected tile by tile. In many cases, this allows that at any given moment, only one tile per MDD has to be kept in server main memory. In some cases, like aggregate operations and predicate evaluation, there is a chance for premature termination of tile stream evaluation.

A number of optimizations (Ritsch, 1999) is applied to a query tree prior to its execution (Widmann, 2000). As an example, consider the query

```
SELECT a < avg_cells(b+c) FROM a, b, c
```

Figure 7 shows the pertaining operator tree where  $op_{ind}$  indicates that operation  $op$  is induced, that is, applied to all cells of the MDD operand. Obviously, such operations are particularly costly. Bold arrows in the tree denote that MDD tiles have to be streamed into the operator, and regular arrows denote scalars (like single numbers) flowing into the operator. The optimization rule applicable in this case, Figure 7, says “averaging over an MDD which is the sum of two MDDs is equivalent to first averaging over the MDDs and then summing.” Figure 7 shows the modified tree; we find that the original tree contained four tile streams while the optimized tree contains only three, yielding a performance gain by 25%.

Figure 7. Original parse tree (left), optimization rule (center), optimized tree (right)



Altogether, rasdaman knows about 150 algebraic re-writing rules; 110 are actually optimizing while the other 40 serve to transform the query into canonical form.

Current work involves parallel evaluation of rasql queries (Hahn & Reiner, 2002), the idea being that tiles form a natural unit for assigning work to different processors. To accommodate arrays larger than disk space, hierarchical storage management (HSM) support is being investigated (Reiner & Hahn, 2002; Sarawagi & Stonebraker, 1994).

### Benchmark Results

The following benchmark (Table 1), based on a real-world GIS scenario, is taken from Baumann (2001); see there for details. They have been obtained on a Pentium III based Linux notebook (650 MHz) with 256 MB main memory. It ran an Oracle 8.1.7 server together with the rasdaman server and the query client, plus the X server. Data sets used were a 36,000×36,000 aerial image (Ortho), a 48,000×47,000 topographical map (1:10,000) containing nine thematic layers (Tk10), and a Digital Elevation Model (Dem). The JPEG images generated by the queries varied between 10 kB and 25 kB in size. Aerial image zoom factors were chosen randomly; no difference was observed in response time.

Table1. Benchmark results for selected real-life queries

Query type	Average elapsed time
SELECT jpeg( scale( Ortho[x0:x1,y0:y1], f ) * {1c,1c,1c} ) FROM Ortho	0.381 sec
SELECT jpeg( scale( bit( Tk10[x0:x1,y0:y1],streets), f ) * {255c,0c,0c} ) FROM Tk10	0.360 sec
SELECT (Dem < 5.0) * {255c,0c,0c} + (Dem > 5.0 AND Dem < 100.0) * {0c,255c,0c} + (Dem > 100.0) * {0c,0c,255c} FROM Dem	0.887 sec

### FUTURE TRENDS

Research on database support for multidimensional raster support is still in its infancy. While basic concepts have been clarified and first reference implementations are available, there is a wealth of open research topics; the following non-exhaustive list indicates some of them:

- unified modeling for statistical and sensor/image databases;
- design methodologies;
- expressiveness of raster languages;
- optimization of complex queries; while there is a large body of research for statistics databases, the field is rather new for sensor/image databases;
- orchestration of MDD with other data types to form comprehensive real-world descriptions;
- improved understanding of tuning parameters;
- standardized benchmarks; and
- best-practice know-how for application modeling in the various application fields.

### CONCLUSION

There is a strong application pull for raster data management in various fields, such as geo-services, life science,



and grid computing. They all have huge assets in common, and research has shown that unified, cross-domain support is feasible. The ultimate goal should be similar to what has been achieved with full-text search and databases: information systems where raster objects are seamlessly embedded in general document structures, raster retrieval forming an integral part of overall retrieval facilities.

## REFERENCES

- Arya, M., Cody, W., Faloutsos, C., Richardson, J., & Toga, J. (1994). QBISM: Extending a DBMS to support 3D medical images. *Proceedings of the 10th International Conference on Data Engineering (ICDE'94)*.
- Baumann, P. (1994). On the management of multidimensional discrete data. *VLDB Journal, Special Issue on Spatial Database Systems*, 4(3), 401-444.
- Baumann, P. (1999). *A database array algebra for spatio-temporal data and beyond*. *Proceedings of the 5th Workshop on Next Generation Information Technologies and Systems (NGITS'99)*, Zikhron-Yaakov, Israel.
- Baumann, P. (2001). Web-enabled raster GIS services for large image and map databases. *Proceedings of the 5th International Workshop on Query Processing and Multimedia Issues in Distributed Systems (QPMIDS 2001)*, Munich, Germany.
- Buneman, P. (1993). *The discrete fourier transform as a database query* (Tech. Rep. No. MS-CIS-93-37). University of Pennsylvania.
- Chock, M., Cardenas, A., & Klinger, (1984). Database structure and manipulation capabilities of a picture database management system (PICDMS). *IEEE ToPAMI*, 6(4), 484-492.
- Furtado, P. (1999). *Storage management of multidimensional arrays in database management systems*. PhD thesis, Technische Universität München.
- Hahn, K., & Reiner, B. (2002). Parallel query support for multidimensional sata: Inter-object parallelism. *Proceedings of the DEXA*, Aix en Provence, France.
- Joseph, T., & Cardenas, A. (1988). PICQUERY: A high level query language for pictorial database management. *IEEE ToSE*, 14(5), 630-638.
- Kleese, K. (2000). A national data management centre. *Proceedings of the 1st International Workshop on Advanced Data Storage/Management Techniques for High Performance Computing*, Daresbury Laboratory, UK.
- Libkin, L., Machlin, R., & Wong, L. (1996). A query language for multidimensional arrays: Design, implementation, and optimization techniques. *Proceedings of the ACM SIGMOD* (pp. 228-239).
- Lorie, R.A. (1982). Issues in databases for design applications. In J. Encarnação & F.L. Krause (Eds.), *File structures and databases for CAD*. North-Holland.
- Marathe, A.P., & Salem, K. (1997). A language for manipulating arrays. *Proceedings of the VLDB'97* (pp. 46-55).
- Marathe, A.P., & Salem, K. (1999). Query processing techniques for arrays. *Proceedings of the ACM SIGMOD '99* (pp. 323-334).
- Reiner, B., & Hahn, K. (2002). Hierarchical storage support and management for large-scale multidimensional array database management systems. *Proceedings of the DEXA*, Aix en Provence, France.
- Ritsch, R. (1999). *Optimization and evaluation of array queries in database management systems*. PhD thesis, Technische Universität München.
- Ritter, G., Wilson, J., & Davidson, J. (1990). Image algebra: An overview. *Computer Vision, Graphics, and Image Processing*, 49(3), 297-331.
- Sarawagi, S., & Stonebraker, M. (1994). Efficient organization of large multidimensional arrays. *Proceedings of the ICDE'94* (pp. 328-336).
- Tamura, H. (1980). Image database management for pattern information processing studies. In F. Chang (Ed.), *Pictorial information systems* (pp. 198-227). LNCS 80. Springer.
- Vandenberg, S., & DeWitt, D. (1991). Algebraic support for complex objects with arrays, identity, and inheritance. *Proceedings of the ACM SIGMOD Conference* (pp. 158-167).
- Widmann, N. (2000). *Efficient operation execution on multidimensional array data*. PhD thesis, Technische Universität München.

## KEY TERMS

**Array:** An item of this information category consists of a nonempty set of (point,value) pairs where the points have n-D integer coordinates and together completely cover an n-D interval, the so-called spatial domain of the array.

**BLOB:** "Binary large object"; a (usually large, i.e., MB to GB) byte string stored in the database; the DBMS does



## **Raster Databases**

not have any knowledge about the semantics of the byte string; hence, it cannot offer semantically adequate functionality and query optimization.

**Hierarchical Storage Management (HSM):** An extension of storage management to include tape media as “tertiary storage”, thereby extending “primary storage” (main memory) and “secondary storage” (disks) by an additional hierarchy level. For management of spatio-temporal objects such as raster data, spatial clustering on tape is an issue to minimize data access and tape load cycles.

**MDD (Multidimensional Discrete Data):** Array.

**Raster Data:** The information category of arrays, also called MDD.

**Raster Database (Management) System:** A DBMS which offers support for storage and retrieval of raster data.

**Tiling:** The technique of decomposing raster data objects into smaller raster items (“tiles”) for storage purposes. Tiling should not be visible to the user but should be resolved internally by the database system.

**Spatial Clustering:** Storage techniques to place data items which have spatio-temporal proximity at neighbored storage locations. Based on the assumption that data access patterns convey some locality behavior, access performance can be improved.

**R**

# Real-Time Databases

**Alejandro Buchmann**

*Technische Universität Darmstadt, Germany*

## INTRODUCTION

A variety of applications in the areas of control, navigation, mobile systems, telecommunications, and simulation depend on the timely and predictable delivery of data. Aircraft and spacecraft control are characterized by small main-memory databases with extremely short mean-latency requirements. Onboard navigation systems tend to combine a small portion of dynamic data with large amounts of static data, e.g., maps and landmark information. The same is true for many simulation environments, e.g., flight simulators or virtual test benches, where some parts of the system are simulated and some parts are physical components. In telecommunications the notion of real time is interpreted mostly statistically, i.e., a predetermined percentage of transactions must meet their deadline. Demanding requirements are imposed by mobile networks with extremely high transaction rates and short response times. A requirements analysis of real-time database systems can be found in Buchmann and Liebig (2001); Locke (1997); Purimetla, Sivasankaran, Ramamritham, and Stankovic (1995); and Raatikainen (1997).

Real-time database systems have two distinct properties that set them aside from conventional general-purpose database systems: The data in the database has time semantics associated with it, and the DBMS must be time-cognizant in order to meet the timing requirements of the transactions. Therefore, the correctness criteria of a real-time database system, in addition to those of conventional databases, include the temporal consistency of data and the time-constrained execution of transactions. A major difference between conventional and real-time databases is the focus on individual transactions. While conventional database systems stress fairness and throughput, real-time systems must ensure that critical transactions are executed in a timely manner. An excellent discussion of common misconceptions about real-time databases is found in Stankovic, Son, and Hansson (1999). Good surveys and background information can be found in Bestavros, Lin, and Son (1997); Ramamritham (1993); and Ramamritham, Son, and DiPippo (2004).

Real-time databases are used in real-time systems that consist of a controlled system, the environment, and a controlling system, the computational system that

coordinates the activities of the controlled system. The state of the environment is typically captured by sensors, whose readings make up a significant portion of the data in a real-time database. For the controlling system to have an accurate view of the state of the controlled system, the monitoring of the environment must occur in a timely fashion. For the actions prescribed by the controlling system to arrive and be executed in time, it is necessary to process the data under timing constraints. Timing constraints in a real-time database system are thus derived both from the limited temporal validity of data and the timing requirements of the environment (Ramamritham, 1993; Ramamritham et al., 2004).

## TEMPORAL CONSISTENCY

Temporal consistency of data has two aspects. *Absolute temporal consistency* defines the consistency between the state of the environment and the data in the database describing the state of the observed physical system at a given time. *Relative temporal consistency* refers to the temporal proximity of data that is used for deriving other data and reflects the fact that measurements of data used together in a computation should be taken within a specified time interval.

All time-sensitive data in a real-time database must carry a time stamp (ts). In addition to the time stamp, data values have a validity interval associated during which data is deemed to be sufficiently fresh to accurately reflect the state of the environment. We call this the absolute validity interval (avi). Absolute temporal consistency is thus defined as a triplet  $d:(value, avi, ts)$ . If at any time  $t$  the difference between  $t$  and  $ts$  is smaller than  $avi$ , the data value has absolute temporal consistency.

If a computation uses data collected by multiple sensors at different times, relative temporal consistency ensures that these readings were not taken too far apart in time and thus ensures that the result is still meaningful. The relative consistency set  $R$  includes all the data involved in a computation.  $R$  has a relative validity interval (rvi) which specifies the maximum permissible difference between the time stamps of the data in  $R$ . For the data in  $R$  to possess relative temporal consistency, the difference between the time stamps of

the data in  $R$  taken pair-wise must be smaller than  $r_{vi}$ . Note that relative temporal consistency is only required for data being used together in deriving other data or some triggering condition for an action.

We can now state that  $d$  in a given relative consistency set  $R$  is temporally consistent if it fulfils the conditions for:

Absolute temporal consistency:  
 $(t - t_s) \leq a_{vi}$

Relative temporal consistency:  
 $\forall d_i \in R, |t_{s_i} - t_{s_j}| \leq r_{vi}$

A real-time database is consistent if it fulfils the normal consistency criteria and is temporally consistent. However, while in a standard database system consistency is preserved by not executing a transaction, in a real-time database consistency must be actively restored since the progress of time causes temporal inconsistency independent of the execution of transactions.

## TIMING CONSTRAINTS

In a real-time database system there are two sources of timing constraints: the freshness requirement of the data given by their respective  $a_{vi}$  and the dynamics of the system that require that computations be available at a given time. For example, a robot navigating a factory floor must have completed its computation (possibly involving data from the real-time database) before it gets to the point at which it must turn.

Absolute validity intervals translate into timing constraints on the sampling process, i.e., the sensor reading must be updated with a frequency proportional to the  $a_{vi}$  (typically  $a_{vi}/2$ ). However, if data is used with other data in a relative consistency set  $R$ , then the highest sampling frequency, i.e. the lowest  $a_{vi}$  in the set  $R$ , must be applied to all data in  $R$ . Similarly, if one data item belongs to multiple relative consistency sets, the lowest  $r_{vi}$  prevails. A detailed discussion on deriving timing constraints from the data's time semantics can be found in Ramamritham (1993). Timing constraints derived from the temporal consistency requirements of the data are usually periodic.

The timing constraints on reactions to a change of state in the system typically are aperiodic. This could be a reaction to an observed movement of a ship. From the application's point of view, the timing constraint specifies the maximum time that may pass before a given (corrective) action is completed, i.e., the deadline. Given that most controlled systems include other components, for example, mechanical components with con-

siderable lag time, we must deduct those time requirements to calculate the available time for computation. From the point of view of a real-time database, we are only interested in the time available for performing database transactions, be these write-only transactions used for writing sensor data, read-only transactions that retrieve data and pass them on to actuators, or update transactions that read from the database, derive new data, and store these back in the database.

Deadlines and the corresponding timing constraints are best characterized through a value function. It indicates the value for the overall system that a task is completed at a given time. If the task or transaction is completed before its deadline, the full value is derived from it. But if the deadline cannot be met, three different situations can arise:

- a smaller positive value is derived from the late execution of a transaction, in which case we speak of soft deadlines and soft real-time constraints;
- no value is derived from late execution, i.e., the value function drops to zero at the deadline;
- a (potentially large) negative value results from late execution; in some cases this is associated with the loss of life and property.

Some authors distinguish between firm and hard deadlines, others consider any transaction without a positive value if completed late to be hard real-time. Within the same system, tasks with soft and hard deadlines can coexist.

Dealing with deadlines raises one important question: How can we determine if a transaction will meet its deadline? This issue has been at the heart of many controversies in the real-time database community. To predict whether a transaction will complete before the deadline or not, it is necessary to know its time of execution. If the real-time database system must give guarantees that a transaction will complete before its deadline, the worst-case execution time is required. If no execution times can be determined, then the only alternative is to evaluate after completion whether the deadline was met or not.

One important difference between conventional and real-time database systems is the fact that in conventional systems, transactions carry no timing information beyond an arrival time stamp. They don't have deadlines and, consequently, execution times of individual transactions are immaterial. Performance of the system is measured by throughput irrespective of individual transactions. Real-time databases, on the other hand, are concerned with the timely execution of individual transactions which have individual deadlines that must be met. Therefore, execution times of individual

transactions are important and necessary for the time-cognizant protocols (schedulers, concurrency control protocols) to work properly.

## DETERMINING EXECUTION TIMES

Execution times depend on a variety of factors (Buchmann, Dayal, McCarthy, & Hsu, 1989). We can dissect the contributions to the total execution time of an application task with an embedded transaction as follows:

$$t_{\text{exec}} = t_{\text{db}} + t_{\text{I/O}} + t_{\text{int}} + t_{\text{appl}} + t_{\text{comm}}$$

where  $t_{\text{db}}$  is the portion of the time consumed by the processing of the database operators, e.g., selections, projections or joins;  $t_{\text{I/O}}$  is the portion spent on I/O, e.g., disk accesses;  $t_{\text{int}}$  is the time due to transaction interference, e.g., waiting for locks;  $t_{\text{appl}}$  is the time spent processing the non-database part of the application code; and  $t_{\text{comm}}$  is the time consumed by communication. To determine worst-case execution times in a deterministic manner, each of these contributing elements must have an upper bound.

The duration of database operations depends on the data. To establish an upper bound for  $t_{\text{db}}$  it is necessary to limit the size of data, e.g., the number of tuples in a relation. This is commonly done in real-time systems since it is also the basis for calculating  $t_{\text{appl}}$ . The path an execution takes also depends on the data. To constrain unpredictability one should avoid recursive or dynamically constructed data structures and unbounded loops.

Determining a reasonable upper bound for  $t_{\text{I/O}}$  is difficult, if not impossible, for disk-resident databases. Assuming for the worst case one page fault for every access, the time estimates become unrealistically large, since disk access time is orders of magnitude slower than main memory access. The result would be that nothing gets scheduled because the bloated worst-case execution time suggests that the deadlines could not be met. Pre-execution—an approach proposed in O’Neil, Ramamritham, and Pu (1996) in which the transaction executes once without acquiring locks to determine what data is needed and to set the buffer, followed by the real execution in a second step—postpones the problem but doesn’t guarantee end-to-end predictability. Any disk-based approach can at best provide statistical values for the expected execution time. In addition, many real-time applications require mean read and write latencies in the sub-millisecond range, something not achievable with disk-resident data. Therefore, the only feasible solution for

systems that must give guarantees is avoiding disk I/O altogether by using main memory database systems. 64-bit address spaces and a drastic drop in memory costs make it feasible to accommodate the structured data needed by real-time applications. Large-volume static data and streaming multimedia data that is accessed without transactional semantics can be accommodated by a hybrid approach with large buffers for data staging.

Time of interference  $t_{\text{int}}$  refers to the time a transaction waits for resources held by another transaction or the time required for rollback and restart due to transaction abort. Disk-based database systems must schedule transactions in parallel in order to maximize the use of resources, such as CPU, while slow accesses to disk are performed. To increase intertransaction parallelism it is necessary to minimize the size of the locking unit, for example, at the tuple level. Since the tuples that are accessed by a transaction may depend on previous operations of the transaction, dynamic lock acquisition has become the method of choice in conventional DBMSs to guarantee small locking granules and increase parallelism. Dynamic lock acquisition necessarily results in scheduling policies based on conflict detection and conflict resolution. The bulk of the real-time database research has followed this approach because the basic assumptions of disk-based systems have never been challenged. However, moving to main memory databases opens new perspectives: Since the data is available in main memory, no lengthy disk accesses are needed, and passing control from one transaction to another with all the overhead of context switching becomes counterproductive. Instead, transactions should be allowed to run with as few interruptions as possible. Since intertransaction parallelism becomes less important, bigger data units, e.g., complete tables, can be locked. This in turn makes it possible to determine the needed resources before execution starts (e.g., by looking at the FROM clause in an SQL statement). This syntactic analysis can be done offline for the transaction classes used in real-time systems. Once data resources are known, a whole new class of conflict-avoiding schedulers becomes feasible (Ulusoy & Buchmann, 1998). The main advantage of these schedulers is that  $t_{\text{int}}$  due to blocking and rollback is eliminated, since a transaction starts execution only when it has acquired all the needed resources. Waiting time in the queue can be predicted from the execution times of the tasks ahead in the queue.

Since  $t_{\text{appl}}$  and  $t_{\text{comm}}$  are not database specific but part of all real-time processes and have been widely analyzed in the real-time community, they are not considered here.

## TRANSACTION PROCESSING

Transaction processing in a real-time database system is quite different from its counterpart in conventional systems: To be effective, the protocols involved must be time-cognizant. This includes the scheduler, the concurrency control mechanism, and the recovery subsystem.

What scheduling policy can be used depends on the availability of timing and resource information. Timing information comprises time of arrival, deadline, a value function, and worst-case execution time. If no execution time is available, then the evaluation of the value function can only be done at arrival time and degenerates to a priority. Resources, from a database point of view, are read sets and write sets.

What protocols work best depends on the transaction workload and its real-time requirements, i.e., whether the transactions arrive periodically or aperiodically, and on the nature of the timing constraints. Often these go hand in hand.

The vast majority of transactions with hard timing constraints are periodic, for example, sensor update transactions, the calculation of derived data, and the reactions when certain threshold conditions are met. While reactions may occur only sporadically, the necessary resources must be reserved in each cycle. A common misconception is that aperiodic transactions can be reduced to periodic transactions using the shortest possible time between two arrivals as the period. This approach only works when arrival times of aperiodic transactions are multiples of the minimum, i.e., it is reduced to the sporadic case discussed above. Totally aperiodic transactions with hard timing constraints usually cannot be handled unless resources are specifically set aside. Those systems that handle aperiodic transactions with hard deadlines only guarantee that such transactions, once scheduled, will be executed to completion (O'Neil et al., 1996; Ulusoy & Buchmann, 1998). Periodic transactions can be scheduled using either a static table-driven or a preemptive priority-driven approach. In the former, time slots are reserved based on worst-case execution times, and if a transaction executes faster, the resources either idle or are used for non-hard deadline tasks. Among the preemptive priority schedulers, the rate monotonic approach (Shah, Rajkumar, & Lehoczky, 1988) is quite popular. It assigns the task with the smallest periodicity the highest priority and works well if periodicity is deterministic and there are no data dependencies among tasks.

Transactions with soft timing constraints have a value function associated that has a positive value after the deadline. However, many approaches reported in the literature use only a degenerate form of the value func-

tion, which is reflected in the metric for success that is used. For example, if the same constant value function is used for all transactions and it is a step function that drops to zero at the deadline, then the success metric is reduced to number of transactions completing before the deadline. Similarly, the scheduling policy that can be used depends on the availability of execution time information for each transaction. If this is not known, the scheduler can only decide based on the deadlines of a transaction. The most popular deadline-based approach is earliest deadline first. If worst case or average execution times are known, other more sophisticated scheduling policies can be applied, such as highest value first, which schedules first the transaction contributing most to the total value of the system; highest value density first, which schedules first the transaction with the highest value per unit of computation time; longest time first, which schedules first the transaction with the longest execution time; or total value, which schedules transactions maximizing the total system value.

A large number of time-cognizant concurrency control mechanisms have been proposed. Here we will comment only on a representative selection. Most of them assume dynamic locking and are therefore conflict resolving. These algorithms vary mainly in the conflict resolution strategy they apply. Priority abort (Abbott & Garcia-Molina, 1992) resolves data conflicts always in favour of high priority transactions. If a lock conflict occurs and the lock-holding transaction has higher priority, the requesting transaction is blocked. Otherwise the lock-holding transaction is aborted. Provided all transactions have different priority, this algorithm is deadlock free. In the priority inheritance protocol (Shah, Rajkumar, & Lehoczky 1990), the problem of a transaction with a higher priority waiting for a lower priority transaction is solved by letting the low priority transaction execute at the priority of the higher priority transaction that is waiting. In a better performing variant of priority inheritance, a conflicting low priority transaction is aborted unless it is close to completion, in which case it inherits the higher priority to finish faster (Shah, Rajkumar, Son, & Chang, 1992). This obviously requires some knowledge of the total execution time. Optimistic Wait-50 (Haritsa, Carey, & Livny, 1990) is an optimistic protocol that executes first and validates before the commit. A finishing transaction is validated against the executing transactions. If half or more of the conflicting transactions have higher priority, then the validating transaction waits; otherwise it may commit and the conflicting transactions are aborted. The pre-claiming algorithm presented in Ulusoy and Buchmann (1998) is a conflict-avoiding protocol that only starts a transaction when it has secured all its resources. An arriving transaction tries to acquire all its resources and



if successful is enqueued in the ready queue. Otherwise it is enqueued in the wait queue. Whenever a transaction commits and frees its resources, the transactions in the wait queue are reexamined and if they complete their resources are transferred to the ready queue. It has been shown that this algorithm outperforms the others discussed above in main memory environments.

Overload management deals with the trade-offs that can be made to ensure timeliness at the expense of another property. Rather than aborting a transaction that can't meet its deadline or refusing to schedule it, often a transaction of lesser cost but lower quality results can be executed. Typical trade-offs are timeliness vs. precision, completeness, currency, or consistency (Hou, Ozsoyoglu, & Taneja, 1989; Ramamritham, 1993).

Recovery is a difficult task that is made more complex in the presence of deadlines. Aborting and undoing the changes of a transaction due to lock conflicts or because they miss their deadline can be a time-consuming task that jeopardizes other transactions that may not even be involved in a lock conflict. It has been observed in conventional database systems that a high percentage of aborted transactions may cause a system to thrash. In real-time systems, where we have, in addition to the aborts due to deadlock, the aborts due to time-constraint violation, the threshold is much lower. These arguments speak in favour of conflict-avoiding protocols. Improved logging mechanisms that use nonvolatile high speed memory are presented in Sivasankaran, Ramamritham, and Stankovic (2001), and logging approaches that treat data differently according to their criticality and achieve bounded recovery time are introduced in Shu, Stankovic, and Son (2002).

## CONCLUSION AND OUTLOOK

Real-time data management is transcending the world of real-time databases. As many applications view data as flowing and the network as the database, real-time data services are gaining interest, particularly in sensor network applications, mobile real-time databases, and Web-based real-time data services. Many of the ideas and insights developed for real-time databases can be adapted to these new areas. A deep understanding of the underlying technologies and their interactions is needed to define the abstractions that will allow us to provide a platform for real-time data services.

## REFERENCES

- Abbott, R., & Garcia-Molina. (1992). Scheduling real-time transactions: A performance evaluation. *ACM Transactions on Database Systems*, 17(3), 513-560.
- Bestavros, A., Lin, K.-J., & Son, S. (1997). *Real-time database systems: Issues and applications*. Boston: Kluwer Academic Publishers.
- Buchmann, A., Dayal, U., McCarthy, D., & Hsu, M. (1989). Time-critical database scheduling: A framework for integrating real-time scheduling and concurrency control. *Proceedings of the Fifth International Conference on Data Engineering* (pp. 470-480).
- Buchmann, A., & Liebig, C. (2001). Distributed, object-oriented, active, real-time DBMSs: We want it all—Do we need them (at) all? *Annual Reviews in Control* (Vol. 25, pp. 147-155).
- Haritsa, J. R., Carey, M. J., & Livny, M. (1990). Dynamic real-time optimistic concurrency control. *Proceedings of the 11th Real-Time Symposium* (pp. 94-103).
- Hou, W.-C., Ozsoyoglu, G., & Taneja, B. K. (1989). Processing aggregate relational queries with hard time constraints. *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data* (pp. 68-77).
- Locke, D. (1997). Real-time databases: Real-world requirements. In A. Bestavros, K. Lin, & S. Song (Eds.), *Real-time database systems: Issues and applications*. Boston: Kluwer Academic Publishers.
- O'Neil, P., Ramamritham, K., & Pu, C. (1996). A two-phase approach to predictably scheduling real-time transactions. In *Performance of concurrency control mechanisms in centralized database systems* (pp. 494-522). Englewood Cliffs, NJ: Prentice Hall.
- Purimetla, B., Sivasankaran, R., Ramamritham, K., & Stankovic, J. A. (1995). Real-time databases: Issues and applications. In S. Son (Ed.), *Advances in real-time systems*. Englewood Cliffs, NJ: Prentice Hall.
- Raatikainen, K. (1997). Real-time databases in telecommunications. In A. Bestavros, K. Lin, & S. Son (Eds.), *Real-time database systems: Issues and applications*. Boston: Kluwer Academic Publishers.
- Ramamritham, K. (1993). Real-time databases. *Distributed and Parallel Databases*, 1, 199-226.

## Real-Time Databases

- Ramamritham, K., Son, S., & DiPippo, L. C. (2004). Real-time databases and data services. *Real Time Systems Journal*, 28, 179-215.
- Shah, L., Rajkumar, R., & Lehoczky, J. P. (1988, March). Concurrency control for distributed real-time databases. *ACM SIGMOD Record* (Vol. 17, pp. 179-215).
- Shah, L., Rajkumar, R., & Lehoczky, J. P. (1990). Priority inheritance protocols: An approach to real-time synchronization. *IEEE Transactions on Computers*, 39(9), 1175-1185.
- Shah, L., Rajkumar, R., Son, S. H., & Chang, C. H. (1992). A real-time locking protocol. *IEEE Transactions on Computers*, 40(7), 793-800.
- Shu, L., Stankovic, J., & Son, S. H. (2002). Achieving bounded and predictable recovery using real-time logging. *IEEE Real Time Technology and Applications Symposium (RTAS'02)*.
- Sivasankaran, R., Ramamritham, K., & Stankovic, J. (2001). System failure and recovery. *Real-Time Database Systems*, (pp. 109-124). Boston: Kluwer Academic Publishers.
- Stankovic, J. A., Son, S. H., & Hansson, J. (1999, June). Misconceptions about real-time databases. *IEEE Computer* (Vol. 32, No. 6, pp. 29-36).
- Ulusoy, O., & Buchmann, A. A real-time concurrency control protocol for main-memory database systems. *Information Systems*, 23(2), 109-125.

# Relational, Object–Oriented and Object–Relational Data Models

Antonio Badia

University of Louisville, USA

## INTRODUCTION

The relational data model is the dominant paradigm in the commercial database market today, and it has been for several years. However, there have been challenges to the model over the years, and they have influenced its evolution and that of database technology. The object-oriented revolution that got started in programming languages arrived to the database area in the form of a brand new data model. The relational model managed not only to survive the newcomer but to continue becoming a dominant force, transformed into the object-relational model (also called extended relational, or universal) and relegating object-oriented databases to a niche product. Although this market has many nontechnical aspects, there are certainly important technical differences among the mentioned data models. In this article I describe the basic components of the relational, object-oriented, and object-relational data models. I do not, however, discuss query language, implementation, or system issues. A basic comparison is given and then future trends are discussed.

## BACKGROUND

In order to facilitate the comparison among the models, I will use the same example throughout the article. The example involves a universe of people; some are professors and some are students. Each professor works at a department, and some professors chair departments. Students have professors as advisors. The exact situation is depicted as entity-relationship (ER) diagram (Chen, 1976) in Figure 1. The notation given denotes that Chairs is a one-to-one relationship, Faculty is a one-to-many relationship, and Teaches is a many-to-many relationship. Also, Phones is a multi-valued attribute and Student and Professor are subclasses of Person. The inverted triangle symbol is used to denote a class/subclass relationship; the reader is warned that different authors use different symbols for this purpose.

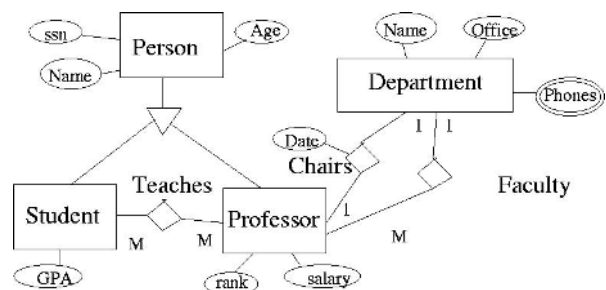
The relational data model (Date, 2004) is well known; one simply overviews its main concepts here as it serves as the baseline for the comparison. A domain is a

nonempty set; intuitively, it provides a pool of values. Every domain is assumed to come with a name (an infinite number of names, technically). Given a schema or list of domain names  $R = \{A_1, \dots, A_n\}$ , a relation on  $R$  is a subset of the Cartesian product  $A_1 \times \dots \times A_n$ . The elements of the relation are called tuples; each tuple is made up of a list of values  $a_1, \dots, a_n$ , with  $a_i$  coming from domain  $A_i$ . A key  $K$  for relation  $r$  in schema  $R$  is a subset of  $R$  (i.e., a set of attributes) such that, for any two tuples in  $r$ , they are the same if they have the same value for  $K$ . Intuitively, the key represents (stands for) the whole tuple, a fact that is exploited in relational database design.

Usually, the domains allowed in most implementations are data types that the computer can easily handle: different numerical types (integers, reals, etc.), characters, and strings. Most database systems also offer a “date” and a “time” domain, to facilitate expression of temporal information as well as *large object* types, frequently used to deal with multimedia data. However, no complex types are allowed. “Complex” here means, roughly, offering the ability to store more than one value. Because tuples will be used to represent entities, this means that attributes with multiple values (or many relationships among objects) will force an object to be represented over several tuples—perhaps even over several tables. This characteristic, called the *first normal form*, will become an issue later on in this article.

A relational-database schema is a set of relation schemas plus a set of integrity constraints. An integrity constraint is a condition specified over the database schema that restricts the data that can be stored in the relations. The most important constraints are the *key*

Figure 1. Entity-relationship diagram for the example



*constraint*, which specifies that certain attribute(s) form a key in a relation, and the *foreign key constraint*, which specifies that certain attribute(s) form a foreign key in a relation. The foreign key attributes  $K1$  have all their values drawn from some primary key  $K2$ ;  $K1$  is said to refer to  $K2$ . Foreign key constraints (also called *referential integrity constraints*) are the glue that holds the relations in a database together by making sure that values in an attribute that need to refer to a certain entity do so. It is therefore necessary to specify, when talking about a relational database, which primary keys and integrity constraints are supposed to hold. Here, as an example, is how our model would be represented in a relational database; for simplicity, I use a stylized syntax, with the relation name first and the attributes as a list in parenthesis. Each foreign key declaration follows the attribute name, and each relation is followed by a primary key declaration:

Person(Ssn, Name, Age); primary key: Ssn.  
Professor(Ssn, Name, Age, Rank, Salary); primary key: Ssn.  
Student(Ssn, Name, Age, GPA); primary key: Ssn.  
Department(Name, Office); primary key: Name.  
Dept-Phone(Name, Phone); primary key: (Name, Phone).  
Faculty(Ssn1 foreign key refers to Professor, Name foreign key refers to Department); primary key: Ssn1.  
Chairs(Ssn1 foreign key refers to Professor, Name foreign key refers to Department, Date), primary key: Ssn1.  
Teaches(Ssn1 foreign key refers to Professor, Ssn2 foreign key refers to Student); primary key: (Ssn1, Ssn2).

Note that each entity gives rise to a table, and each relationship does, too. There is another option in the translation, affecting one-to-many and one-to-one relationships. A one-to-many relationship, such as Faculty, could be represented in the table corresponding to the entity in the one side (in this case, Professor); and so could one-to-one relationships; in fact, in one-to-one relationships there is a choice of tables). Many-to-many relationships need their own separate tables. Also, the multivalued attribute “Phones” cannot be added to the Department table for the reasons already cited (first normal form).

Because having several tables repeating all department information, and each with a different phone, would create redundancy, a separate table is created, with the department name representing the whole department (because Name is the primary key of Department). This is the process of normalization, on which relational database design is based. As for the class/

subclass relation, there is no direct facility to capture it in relational databases; it must be simulated by one of two methods. Both methods have one table for the superclass, and one for each of the subclasses. The first method puts, on the tables corresponding to the subclasses, the attributes proper of the subclass only; the second method combines, on the tables corresponding to the subclasses, the attributes of the subclass and the superclass. I have chosen this second option on the database. Note that the table corresponding to the superclass is still needed in the second method in case there are elements that are objects of the superclass only and not of any subclasses.

## MAIN THRUST: ADVANCED DATA MODELS

### The Object-Oriented Data Model

There are many variations of the object-oriented data model. In this article, we use the object data management group (ODMG) model (Catell et al., 2000) and its object database language (ODL), because they set the standard data model for object-oriented databases.

The basic building blocks of the ODMG data model are *objects* and *literals*. A literal’s value may be simple or complex. There are three types of literals: atomic, collection, and structured. Atomic literals correspond to basic data types: integers (long, short, unsigned), float (float, double), Boolean, single character, string, and enumeration types. Structured literals have a tuple structure; they include types such as Date, Time,... Also, the user can define structured literals as needed, using a “Struct” construct. Collection literals specify a collection of objects or literals. Types of collections are Set, Bag, List (homogeneous), Array, and Dictionary. Each collection has a group of built-in operators. On the other hand, objects have object identifiers (OIDs) and a value, unlike literals, which have value but no OID. Objects may have a name and can be of atomic or collection type. Atomic objects are not objects without internal structure; they correspond to atomic or structured literals. For each object, properties (i. e., attributes and relationships) and operations are specified. Values of attributes are typically literals (atomic or complex) but can be OIDs. Values of relationships are always object names or a collection applied to object names.

Object definition language (ODL) is used to implement the ODMG data model. In ODL, classes are declared by giving them an interface, using the keyword *interface*; an interface declares the structure of an

object and it is noninstantiable (i.e., there cannot be objects instantiating the interface). ODL uses “class” to refer to instantiable declarations. Each class is a collection of attributes that are either primitive (i.e., of a basic data type) or relationship (i.e., their value is an object or set of objects of a given class). A key for a class can be optionally declared. Note that all objects have object identifiers, which are immutable, regardless of whether they have keys (whose values can be changed) or not. Obviously, inheritance among classes is supported directly in the model, and declared using the keyword *extends*; however, only single inheritance is allowed. In the ODMG model, only binary relationships are explicitly represented through a pair of inverse references (using the *inverse* keyword). An example of class declarations can be seen in the following section, implementing the model in Figure 1. Because only binary relationships without attributes can be directly represented in ODL, binary relations with attributes and *n*-ary relationships ( $n > 2$ ) must be represented as classes themselves, a process called *reification*.

```

Class Person {
attribute int ssn;
attribute string name;
attribute int age;
}
Class Professor extends Person {
attribute int rank;
attribute real salary;
relationship Department dept inverse faculty;
relationship set <Student> teaches inverse taught;
relationship Chairs chairing inverse chairp;
}
Class Student extends Person {
attribute float GPA;
relationship Department department;
relationship taught set <Professor> inverse teaches;
}
Class Department {
attribute string name;
attribute string office;
attribute set <string> phones;
relationship Chairs chairperson inverse chairsd;
relationship set <Professor> faculty inverse dept;
}
Class Chairs {
attribute date Date;
relationship Department chairsd inverse chairperson;
relationship Professor chairsp inverse chairing;
}

```

Note that only binary relationships without attributes can be nicely captured in the model. A binary relationship with attributes (Chairs) must be reified (i.e., converted

to a class) in order to be represented. The same would happen to relationships that involved more than two entities. On the other hand, complex or multivalued attributes are not a problem; they can be easily represented using structures or sets, so that each entity in our ER model corresponds to a class.

## The Object-Relational Data Model

The object-relational (also called extended-relational, or universal) data model is exemplified by the latest version of the Structured Query Language or SQL-99 (Melton, 1999). It can be seen as an attempt to capture as many as possible of the object-oriented concepts introduced in the previous subsection and wrap them in a relational shell. A more modest view of it regards the model as extending the base of the relational model (instead of the model itself) by making it easier to add more complex data types to serve as domain definitions. Here the basics of the standard (Eisenberg, Kulkarni, Michaels, Melton, & Zemke, 2004) are described, because each commercial DBMS has its own version of the model, sometimes with different names and different syntax.

One of the basic ideas is to substitute domains by (possibly complex) types, called user defined types (UDTs). The name comes from the fact that the model provides constructors so that users can define their own types as needed by an application; the emphasis is in extendibility. UDTs come in *distinct* types and *structured* types. Distinct types are based on a single, built-in data type. The following is an example of an UDT called “age” based on the built-in type integer:

```

CREATE TYPE age AS INTEGER (CHECK age BETWEEN 0 and 100) FINAL;

```

Distinct types are not compatible with any other type (including the one they are based on); hence, expressions such as “age + 20” or “age > 7” are illegal operations (however, the CAST operator can be used as a work around: “CAST(age AS INTEGER) + 20” is allowed.). The keyword *final* refers to whether a type can be extended with subtypes: distinct types cannot, structured ones can. As an option, operations and comparisons can be defined for types. Distinct types are first class: they can be used in a column declaration, SQL variable, and so forth. Here is a table where the above declaration is used for a column:

```

CREATE TABLE person (
Ssn INTEGER,
Name CHARACTER VARYING(100),
Age age)

```



Structured types can have internal structure, with parts called attributes. Attributes do not have to be built-in; they may be complex (SQL99 offers “built-in” structured types ARRAY and ROW), but cannot be of the type being defined (i.e., recursion is not allowed). Structured types do not have identity. One cannot specify constraints on the attributes of a structured type. A definition can be destroyed by using the command DROP TYPE; it can also be changed with command ALTER TYPE, but there are restrictions on what can be changed. For instance, it is not allowed to add or delete attributes if the type being changed is a supertype, an element of another type, or a reference (see the following section for an explanation of references).

Structured types can be used as columns of tables and also as tuples of tables. For example, one could have defined a structured type as follows instead of the previous table:

```
CREATE TYPE person as (
  Ssn INTEGER,
  Name CHARACTER VARYING(100),
  Age age)
And then use the type as a row:
CREATE TABLE Persons OF person
(REF IS person-id SYSTEM GENERATED)
```

(I will explain REF shortly). This yields a *typed table*, a table whose rows are of a structured type. The attributes of the type become attributes of the table.

Structured types can have methods defined on them. An *observer* and a *mutator* are methods automatically created for every attribute; they provide encapsulation. A *creator* method is also defined, which is invoked by NEW. Types may be NOT INSTANTIABLE (i.e., cannot have values of that type; this is used for abstract super-classes) or INSTANTIABLE.

It is possible to define a hierarchy of types by defining a new type UNDER another; (single) inheritance applies:

```
CREATE TYPE student UNDER person AS (
  GPA DECIMAL(1,2),
  Department REF(department))
```

Typed tables also have inheritance hierarchies, corresponding to the hierarchies of the types:

```
CREATE TABLE Students of Student;
(Department WITH OPTION SCOPE Departments)
```

Subtables cannot have primary keys, but supertables can. They can also have UNIQUE, NOT NULL, CHECK and integrity constraints. There is also a self-reference

value created automatically, unique for each row of the table. SQL-99 provides REFERENCE types, which give structured types an ID. Maximal supertables must specify the self-referencing column (it is inherited by subtables, which cannot specify such a column on their own). References can be generated by system, by using some built-in type, or by using some attributes of the type. The declaration of table Students specifies that the reference (REF) to department on each student must be to a valid reference of departments within the table Departments, which is a table of Department types.

SQL-99 introduces also *typed views*. Typed views can have a hierarchy, also.

We now show the rest of the declarations needed to complete the modeling of Figure 1 in the object-relational data model:

```
CREATE TYPE department (
  Name CHARACTER VARYING(100),
  Office CHARACTER VARYING(100),
  Phones CHARACTER(10) ARRAY[5],
  Chairperson REF(Faculty))
```

```
CREATE TYPE Professor UNDER person AS (
  Rank INTEGER,
  Salary DECIMAL(5,2),
  Department REF(department),
  Chairs REF(department) )
```

```
CREATE TYPE department (
  Name CHARACTER VARYING(100),
  Office CHARACTER VARYING(100),
  Phones CHARACTER(10) ARRAY[5],
  Chairperson REF(Professor))
```

```
CREATE TABLE Faculty of Professor UNDER Persons
(Chairs WITH OPTION SCOPE Departments)
```

```
CREATE TABLE Departments of DEPARTMENT
(REF IS dept-id SYSTEM GENERATED
Chairperson WITH OPTIONS SCOPE Faculty)
```

```
CREATE TABLE Teaches(
  Teacher REF(Professor) SCOPE Faculty,
  Pupil REF(Student) SCOPE Students))
```

A few observations are in order: First, we could have created tables of persons (or faculty, or students, or departments) without first declaring a type, by giving each row in the table the appropriate attributes (as in the first example of table PERSON); this results in tables similar to those in the original relational model. However, that option would not have allowed us to model type/subtype relationships, and to have inheritance. Sec-

ond, binary relations that are one-to-one (or one-to-many) can be represented through references, similar to ODL. In this case, we add the SCOPE clause to limit references to objects appearing in a table. However, there is no concept of inverse here; it is up to the user to make sure that the references in, for instance, Chairs and Chairperson, are consistent. Note that there is no need for both relations; having Chairs in Professor would be enough to establish the relationship—and indeed, that is how it would be truly modeled; we have added Chairperson to Department for illustration only. However, note that to find the chairperson of a given department could be difficult and costly without this link. Finally, note that many-to-many relationships (and relationships that are  $n$ -ary,  $n > 2$ ) must be modeled in a manner similar to the original relational model, the only difference being that references (instead of primary keys and foreign keys) can be used to refer to the original objects.

## Comparison

For a meaningful comparison among the models introduced, focus on the fundamental aspect of a data model: modeling power. Ask, How does each model deal with the basic constructs of most conceptual models (entities or classes, and relationships among them)? Is there something that is expressible in one model and not in the others? How easy (or intuitive or natural) is it to model complex situations?

The previous analysis makes it possible to answer the question. In theory, there are situations that can be modeled in the object-oriented data model and that do not have a good representation in the relational data model (albeit they can be modeled with some ingenuity). In particular, set-valued attributes are tricky to handle in the relational model and lead to dissemination of related information in several tables; also, type/subtype relations and inheritance must be simulated. In practice, however, the limitations of object-oriented models to deal with relationships (especially relationships with attributes and relationships that are many-to-many) mean that neither model can claim a significant advantage. The object-relational model, offering some of the abilities of object-oriented models (inheritance, identifiers) and relational models (clear modeling of relationships) may claim a small advantage over both of them. However, the model as defined by the SQL standard (Melton, 1999) still lacks the ability to handle sets well. It seems that the last iteration of the standard removes this hurdle, in which case the object-relational model could become the winner in terms of modeling power. However, no model is able to handle semistructured

or irregular data very well (Abiteboul, Buneman, & Suciu, 1999; Mani & Badia, 2003).

## FUTURE TRENDS

New developments in the area of data models have been directed towards capturing data in semistructured form (XML; Abiteboul, Buneman, & Suciu, 1999). Integration of XML with relational data will continue to attract researchers in coming years, because at present such integration is not well defined. It is interesting to point out that this issue has revived the interest in *nested relations*, an old data model (Paredaens, De Bra, Gyssens, & Van Gucht, 1989) that continues seeing new uses (Lee, Mani, Chiu, & Chu, 2001) because it is a hierarchical model that is well suited to represent XML data. Another area of interest will be the integration of unstructured information (text), since it is acknowledged that documents (e-mails, memos, technical reports, etc.) are a very important source of information that is currently outside the realm of databases. Current efforts in this direction are based on information retrieval techniques (Baeza-Yates & Ribeiro, 1999), but this approach has limitations that will keep researchers looking for other approaches. Finally, integration of multimedia data (e.g., video, audio, images), which was a big driving force behind the development of extensions of the relational model in the first place, is still an open area of inquiry. Trends like geographical information systems (GIS; Rigaux, Scholl, & Voisard, 2001) that rely heavily on such data will continue to fuel research in this area.

## CONCLUSION

I have presented the basic characteristics of the (pure) relational, the object-oriented and object-relational data models. By showing them side by side, with a common example, it is possible to compare the models easily, a comparison that is absent from the literature. Note that the focus is on conceptual aspects of the model, and not query languages, implementation, and performance issues. The comparison of data models shows that is very difficult to establish a clear winner; each model may be better for a particular type of data. For data modeling capabilities, the object-relational model may offer somewhat more flexibility than the other two. This is not unexpected if this model was created in an attempt to integrate the best of the relational and object-oriented paradigm.

## REFERENCES

- Abiteboul, S., Buneman, P., & Suciu, D. (1999) *Data on the Web: From relations to semistructured data and XML*. San Francisco: Morgan Kaufman.
- Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval*. New York: ACM Press.
- Catell, R. G. (1994). *Object data management*. Reading, MA: Addison-Wesley.
- Catell, R. G., Barry, D., Berler, M., Eastman, J., Jordan, D., Rusell, C., Schadow, O., et al. (2000). *The object data standard: ODMG 3.0*. San Diego, CA: Academic Press.
- Chen, P. (1976). The entity-relationship model - towards a unified view of data. *ACM Transactions on Database Systems*, 1(1).
- Date, C.J. (2004). *An introduction to database systems*. Reading, MA: Addison-Wesley.
- Eisenberg, J., Kulkarni, K., Michaels, J.-E., Melton, J., & Zemke, F. (2004, March). SQL: 2003 has been published. *SIGMOD Record*, 33(1), 119-126.
- Lee, D., Mani, M., Chiu, F., & Chu, W. (2001). Nesting-based relational-to-XML schema translation. *Proceedings of the 4th International Workshop on the Web and Databases, WebDB 2001*, Santa Barbara, CA, May 24-25, in conjunction with ACM PODS/SIGMOD 2001, informal proceedings.
- Mani, M., & Badia, A. (2003, October). Data modeling using XML. Tutorial in the 22<sup>nd</sup> *International Conference on Conceptual Modeling*, Computer Science 2813. New York: Springer.
- Melton, J., & Simon, A. R. (2002). *SQL 1999: Understanding relational language components*. San Francisco: Morgan Kaufmann.
- Paredaens, J., De Bra, P., Gyssens, M., & Van Gucht, D. (1989). *The structure of the relational database model*. New York: Springer.
- Rigaux, P., Scholl, M., & Voisard, A. (2001). *Spatial databases: With application to GIS*. San Francisco: Morgan Kaufmann.

## KEY TERMS

**Inheritance:** A special relation between two classes of objects; class A inherits from class B when it is considered that A possesses all the characteristics of B (and possibly some more) simply because it is a subclass of class B. Models that support inheritance explicitly allow economical expression of facts, because declaring A as a subclass of B implies inheritance in such models.

**Object Identifier:** In the object-oriented data model, each object is given an object identifier (OID). The importance of OIDs is that it makes the model work by reference and not by value (i.e., if an object changes the values of all its attributes, it still is the same object because of its OID). In the relational world, an object that changes the value of one attribute is a different object. Unlike keys, OIDs are immutable.

**Relation:** From a type point of view, a relation is simply a set of tuples. In most implementations of the relational model, however, multisets (sets with repetitions) are used.

**Set:** From a type point of view, a set is a collection construct that is homogeneous (i.e., it contains objects of the same type) and unbounded (i.e., it has no maximum or fixed size). Sets can only be handled as relations in the relational model, but the set constructor in the object-oriented data model means that a much higher degree of flexibility is needed in organizing information.

**User-Defined Type (UDT):** Any type as defined through the use of some basic constructors, such as CREATE TYPE. The object-relational model provides UDT in an attempt to make the system more customizable for applications with particular requirements of the data representation.

# Repairing and Querying Databases with Integrity Constraints

**Sergio Greco**

*DEIS Università della Calabria, Italy*

**Ester Zumpano**

*DEIS Università della Calabria, Italy*

## INTRODUCTION

Data integration aims at providing a uniform integrated access to multiple heterogeneous information sources, which were designed independently for autonomous applications and whose contents are strictly related.

There are several ways to integrate databases or possibly distributed information sources, but whatever integration architecture we choose, the heterogeneity of the sources to be integrated causes subtle problems. In particular, the database obtained from the integration process may be inconsistent with respect to integrity constraints; that is, one or more integrity constraints are not satisfied. The following example shows a case of inconsistency.

**Example 1.** Consider the following database schema consisting of the single binary relation *Teaches* (*Course*, *Professor*) where the attribute *Course* is a key for the relation. Assume there are two different instances for the relations *Teaches*,  $D1 = \{(c1,p1), (c2,p2)\}$  and  $D2 = \{(c1,p1), (c2,p3)\}$ .

The two instances satisfy the constraint that *Course* is a key, but from their union, we derive a relation which does not satisfy the constraint since there are two distinct tuples with the same value for the attribute *Course*.

Obtaining consistent information from inconsistent databases is a primary issue in database management systems. In the integration of two conflicting databases, simple solutions could be based on the definition of preference criteria such as a partial order on the source information or a majority criteria (Lin & Mendelson, 1996). However, these solutions are not generally satisfactory, and more useful solutions are those based on (1) the computation of repairs for the database and (2) the computation of consistent answers (Arenas, Bertossi & Chomicki, 1999). The computation of repairs is based on the definition of minimal sets of insertion and deletion operations so that the resulting database satisfies all constraints. The computation of consistent an-

swers is based on the identification of tuples satisfying integrity constraints and on the selection of tuples matching the goal. For instance, for the integrated database of *Example 1*, we have two alternative repairs consisting in the deletion of one of the tuples  $(c2,p2)$  and  $(c2,p3)$ . The consistent answer to a query over the relation *Teaches* contains the unique tuple  $(c1,p1)$  so that we do not know which professor teaches course *c2*.

Therefore, it is very important, in the presence of inconsistent data, to compute the set of consistent answers, but also to know which facts are unknown and if there are possible repairs for the database.

## BACKGROUND

A (disjunctive Datalog) rule  $r$  is a clause of the form:

$$A_1 \vee \dots \vee A_k \leftarrow B_1, \dots, B_m, \text{not } B_{m+1}, \dots, \text{not } B_n, k+m+n > 0$$

where  $A_p, \dots, A_k, B_p, \dots, B_n$  are atoms of the form  $p(t_p, \dots, t_n)$ ,  $p$  is a predicate symbol of arity  $h$ , and the terms  $t_p, \dots, t_n$  are constants or variables. The disjunction  $A_1 \vee \dots \vee A_k$  is the *head* of  $r$ , while the conjunction  $B_1, \dots, B_m, \text{not } B_{m+1}, \dots, \text{not } B_n$  is the *body* of  $r$ . We also assume the existence of the binary built-in predicate symbols (comparison operators) which can only be used in the body of rules.

An *extended Datalog program* extends standard Datalog programs with a different form of negation, known as classical or strong negation, which can also appear in the head of rules. An extended atom is either an atom, say  $A$  or its negation  $\neg A$ . An extended Datalog program is a set of rules of the form:

$$A_1 \vee \dots \vee A_k \leftarrow B_1, \dots, B_m, \text{not } B_{m+1}, \dots, \text{not } B_n, k+m+n > 0$$

where  $A_p, \dots, A_k, B_p, \dots, B_n$  are extended atoms.

The semantics of an extended program  $P$  is defined by considering each negated predicate symbol, say  $\neg p$ ,



as a new symbol syntactically different from  $p$  and by adding to the program, for each predicate symbol  $p$  with arity  $n$  the constraint  $\leftarrow p(X_1, \dots, X_n), \neg p(X_1, \dots, X_n)$ .

## INTEGRITY CONSTRAINTS

Integrity constraints express semantic information over data, that is, relationships that must hold among data in the theory. Generally, integrity constraints, denoted as IC, represent the interaction among data and define properties which are supposed to be explicitly satisfied by all instances over a given database schema. Therefore, they are mainly used to validate database transactions.

**Definition 1.** A full (or universal) integrity constraint is a formula of the first order predicate calculus of the form:

$$(\forall X) [ B_1 \wedge \dots \wedge B_n \wedge \varphi \supset A_1 \wedge \dots \wedge A_m \wedge \psi_1 \wedge \dots \wedge \psi_k ]$$

where  $A_1, \dots, A_m, B_1, \dots, B_n$  are base positive literals,  $\varphi, \psi_1, \dots, \psi_k$  are built-in literals,  $X$  denotes the list of all variables appearing in  $B_1, \dots, B_n$  and it is supposed that variables appearing in  $A_1, \dots, A_m, \varphi, \psi_1, \dots, \psi_k$  also appear in  $B_1, \dots, B_n$ .

In the definition above, the conjunction  $B_1 \wedge \dots \wedge B_n \wedge \varphi$  is called the *body* and the disjunction  $A_1 \wedge \dots \wedge A_m \wedge \psi_1 \wedge \dots \wedge \psi_k$  the *head* of the integrity constraint. Moreover, an integrity constraint is said to be *positive* if no negated literals occur in it.

## TECHNIQUES FOR QUERYING AND REPAIRING DATABASES

Recently, there have been several proposals considering the integration of databases as well as the computation of queries over inconsistent databases. Most of the techniques work for restricted form of constraints and only recently have there been proposals to consider more general constraints. In the following, we give an informal description of the main techniques proposed in the literature.

- In Agarwal et al. (1995), it is proposed an extension of relational algebra, called *flexible algebra*, to deal with data having tuples with the same value for the key attributes and conflicting values for the other attributes. The technique only considers constraints defining functional dependencies, and it is sound only for the class of databases having dependencies determined by a primary key consist-

ing of a single attribute.

- In Dung (1996), it is proposed that the *Integrated Relational Calculus*, an extension of flexible algebra for other key functional dependencies based on the definition of *maximal consistent subsets* for a possibly inconsistent database. Dung proposed extending relations by also considering null values denoting the absence of information with the restriction that tuples cannot have null values for the key attributes. The Integrated Relational Calculus overcomes some drawbacks of the flexible relational algebra. Anyhow, as both techniques consider restricted cases, the computation of answers can be done efficiently.
- In Lin and Mendelzon (1996), an approach is proposed taking into account the majority view of the knowledge bases in order to obtain a new relation which is consistent with the integrity constraints. The technique proposes a formal semantics to merge first-order theories under a set of constraints.

**Example 2.** Consider the following three relation instances which collect information regarding author, title, and year of publication of papers:

- Bib1 = {(John, T1, 1980), (Mary, T2, 1990)},
- Bib2 = {(John, T1, 1981), (Mary, T2, 1990)},
- Bib3 = {(John, T1, 1981), (Frank, T3, 1990)}

From the integration of the three databases Bib1, Bib2, and Bib3, we obtain the database  $Bib = \{(John, T1, 1980), (Mary, T2, 1990), (Frank, T3, 1990)\}$ .

Thus, the technique, proposed by Lin and Mendelzon, removes the conflict about the year of publication of the paper T1 written by the author John observing that two of the three source databases that have to be integrated store the value 1980; thus, the information that is maintained is the one which is present in the majority of the knowledge bases.

However, the “merging by majority” technique does not resolve conflicts in all cases since information is not always present in the majority of the databases, and therefore, it is not always possible to choose between alternative values. Thus, generally, the technique stores disjunctive information, and this makes the computation of answers more complex (although the computation becomes efficient if the “merging by majority” technique can be applied); moreover, the use of the majority criteria involves discarding inconsistent data, hence, the loss of potentially useful information.



- In Arenas et al. (1999), they introduce a logical characterization of the notion of a consistent answer in a possibly inconsistent database. The technique is based on the computation of an equivalent query  $T_w(Q)$  derived from the source query  $Q$ . The definition of  $T_w(Q)$  is based on the notion of residue developed in the context of semantic query optimization.

More specifically, for each literal  $B$  appearing in some integrity constraint, a residue  $Res(B)$  is computed. Intuitively,  $Res(B)$  is a universal quantified first order formula which must be true because of the constraints if  $B$  is true. Universal constraints can be rewritten as denials, that is, logic rules with empty heads of the form  $\leftarrow B_1 \wedge \dots \wedge B_n$ .

Let  $A$  be a literal,  $r$  a denial of the form  $\leftarrow B_1 \wedge \dots \wedge B_n$ ,  $B_i$  (for some  $1 \leq i \leq n$ ), a literal unifying with  $A$ , and  $q$  the most general unifier for  $A$  and  $B_i$  such that variables in  $A$  are used to substitute variables in  $B_i$ , but they are not substituted by other variables. Then, the residue of  $A$  with respect to  $r$  and  $B_i$  is:

$$\begin{aligned} Res(A,r,B_i) &= \text{not}((B_1 \wedge \dots \wedge B_{i-1} \wedge B_{i+1} \wedge \dots \wedge B_n) \theta) \\ &= \text{not } B_1 \theta \vee \dots \vee \text{not } B_{i-1} \theta \vee \text{not } B_{i+1} \theta \\ &\quad \vee \dots \vee \text{not } B_n \theta. \end{aligned}$$

The residue of  $A$  with respect to  $r$  is  $Res(A,r) = \bigwedge_{B_i|A=B_i\theta} Res(A,r,B_i)$  consisting of the conjunction of all the possible residues of  $A$  in  $r$  whereas the residue of  $A$  with respect to a set of integrity constraints  $IC$  is  $Res(A) = \bigwedge_{r \in IC} Res(A,r)$ .

Thus, the residue of a literal  $A$  is a first order formula which must be true if  $A$  is true. The operator  $T_w(Q)$  is defined as follows:

$$\begin{aligned} T_0(Q) &= Q; \\ T_i(Q) &= T_{i-1}(Q) \wedge R \text{ where } R \text{ is a residue of some literal} \\ &\text{in } T_{i-1}. \end{aligned}$$

The operator  $T_w$  represents the fixpoint of  $T$ . Example 3 is defined by the following first order formula:

$$\forall(X, Y, Z) [ Supply(X,Y,Z) \wedge Class(Z,t) \supset X=c1 ]$$

stating that only supplier  $c1$  can supply items of type  $t$ .

The database  $D = \{ Supply(c1, d1, i1), Supply(c2, d2, i2), Class(i1, t), Class(i2, t) \}$  is inconsistent because the integrity constraint is not satisfied (an item of type  $t$  is also supplied by supplier  $c2$ ).

This constraint can be rewritten as:

$\leftarrow Supply(X,Y,Z) \wedge Class(Z,t) \wedge X \neq c1$ , where all variables are (implicitly) universally quantified. The residue of the literals appearing in the constraint are:

$$\begin{aligned} Res(Supply(X,Y,Z)) &= \text{not } Class(Z,t) \vee X = c1 \\ Res(Class(Z,t)) &= \text{not } Supply(X,Y,Z) \vee X = c1 \end{aligned}$$

The iteration of the operator  $T$  to the query goal  $Class(Z,t)$  gives:

$$\begin{aligned} T_0(Class(Z,t)) &= Class(Z,t), \\ T_1(Class(Z,t)) &= Class(Z,t) \wedge (\text{not } Supply(X,Y,Z) \wedge X = c1), \\ T_2(Class(Z,t)) &= Class(Z,t) \wedge (\text{not } Supply(X,Y,Z) \wedge X = c1), \end{aligned}$$

At Step 2, a fixpoint is reached since the literal  $Class(Z,t)$  has been “expanded”, and the literal  $not Supply(X,Y,Z)$  does not have a residue associated to it. Thus, to answer the query  $Q = Class(Z,t)$  with the above integrity constraint, the query  $T_w(Q) = Class(Z,t) \wedge (\text{not } Supply(X,Y,Z) \vee X = c1)$  is evaluated. The computation of  $T_w(Q)$  over the above database gives the result  $Z=i1$ .

The technique, more general than the previous ones, has been shown to be complete for universal binary integrity constraints and universal quantified queries. However, the rewriting of queries is complex since the termination conditions are not easy to detect, and the computation of answers is generally not guaranteed to be polynomial.

- In Arenas, Bertossi, and Chomicki (2000), they propose an approach consisting in the use of a Logic Program with Exceptions (LPe) for obtaining consistent query answers. An LPe is a program with the syntax of an extended logic program (ELP); that is, in it we may find both logical (or strong) negation ( $\neg$ ) and procedural negation (not). In this program, rules with a positive literal in the head represent a sort of general default, whereas rules with a logically negated head represent exceptions. The semantic of an LPe is obtained from the semantics for ELPs by adding extra conditions that assign higher prior-

Figure 1. Example 3-Consider a database D consisting of the following two relations

Supplier	Department	Item	Item	Type
c1	d1	i1	i1	t
c2	d2	i2	i2	t

ity to exceptions. The method, given a set of integrity constraints (ICs) and an inconsistent database instance, consists in the direct specification of database repairs in a logic programming formalism. The resulting program will have both negative and positive exceptions, strong and procedural negations, and disjunctions of literals in the head of some of the clauses; that is, it will be a disjunctive extended logic program with exceptions. As in Arenas et al. (1999), the method considers a set of integrity constraints written in the standard format  $\bigvee_{i=1}^n P_i(x_i) \vee \bigvee_{i=1}^m (\neg Q_i(y_i)) \vee \varphi$  where  $\varphi$  is a formula containing only built-in predicates, and there is an implicit universal quantification in front. This method specifies the repairs of the database  $D$  that violate IC by means of a logical program with exceptions,  $\Pi^D$ . In  $\Pi^D$  for each predicate  $P$ , a new predicate  $P'$  is introduced, and each occurrence of  $P$  is replaced by  $P'$ . More specifically,  $\Pi^D$  is obtained by introducing:

**Persistence Defaults.** For each base predicate  $P$ , the method introduces the persistence defaults:

$$\begin{aligned} P'(x) &\leftarrow P(x), \\ \neg P'(x) &\leftarrow \text{not } P(x). \end{aligned}$$

The predicate  $P'$  is the repaired version of the predicate  $P$ , so it contains the tuples corresponding to  $P$  in a repair of the original database.

**Stabilizing Exceptions.** From each IC and for each negative literal  $\text{not } Q_{i0}$  in IC, the negative exception clause is introduced:

$$\neg Q'_{i0}(y_{i0}) \leftarrow \bigwedge_{i=1..n} \neg P'_i(x_i), \bigwedge_{i \neq i0} Q'_i(y_i), \varphi'$$

where  $\varphi'$  is a formula that is logically equivalent to the logical negation of  $\varphi$ . Similarly, for each positive literal  $P_{i1}$  in the constraint, the positive exception clause:

$$P'_{i1}(x_{i1}) \leftarrow \bigwedge_{i \neq i1} \neg P'_i(x_i), \bigwedge_{i=1..m} Q'_i(y_i), \varphi$$

is generated. The meaning of the stabilizing exceptions is to make the ICs be satisfied by the new predicates. These exceptions are necessary but not sufficient to ensure that the changes the original subject should be subject to, in order to restore consistency, are propagated to the new predicates.

**Triggering Exceptions.** From the IC in standard form, the disjunctive exception clause:

$$\bigvee_{i=1..n} P'_i(x_i) \vee \bigvee_{i=1..m} Q'_i(y_i) \leftarrow \bigwedge_{i=1..n} \text{not } P_i(x_i), \bigwedge_{i=1..m} Q_i(y_i), \varphi'$$

is produced.

The program  $\Pi^D$  constructed as shown above is a “disjunctive extended repair logic program with exceptions for the database instance  $D$ ”. In  $\Pi^D$  positive defaults are blocked by negative conclusions, and negative defaults by positive conclusions.

**Example 4.** Consider the database  $D = \{p(a), q(b)\}$  with the inclusion dependency  $ID: p(X) \supset q(X)$

In order to specify the database repairs, the new predicates  $p'$  and  $q'$  are introduced. The resulting repair program has four default rules expressing that  $p'$  and  $q'$  contain exactly what  $p$  and  $q$  contain, respectively:

$$\begin{aligned} p'(x) &\leftarrow p(x); \\ q'(x) &\leftarrow q(x); \\ \neg p'(x) &\leftarrow \text{not } p(x); \text{ and} \\ \neg q'(x) &\leftarrow \text{not } q(x); \end{aligned}$$

two stabilizing exceptions:

$$\begin{aligned} q'(x) &\leftarrow p'(x); \\ \neg p'(x) &\leftarrow \neg q'(x); \end{aligned}$$

and the triggering exception:

$$\neg p'(x) \vee q'(x) \leftarrow p(x), \text{not } q(x).$$

The answer sets are  $\{p(a), q(b), p'(a), q'(b), \neg p'(a)\}$  and  $\{p(a), q(b), p'(a), q'(b), q'(b)\}$  that correspond to the two expected database repairs.

The method can be applied to a set of domain-independent binary integrity constraints IC; that is, the constraint can be checked w.r.t. satisfaction by looking to the active domain, and two literals appear in each IC at most.

- In Greco and Zumpano (2000), a general framework for computing repairs and consistent answers over inconsistent databases with universally quantified variables was proposed. The technique is based on the rewriting of constraints into extended disjunctive rules with two different forms of negation (negation as failure and classical negation). The disjunctive program can be used for two different purposes: compute repairs for the database and produce consistent answers, that is, a maximal set of atoms which do not violate the constraints. The technique is sound and complete

(each stable model defines a repair, and each repair is derived from a stable model) and more general than techniques previously proposed.

More specifically, the technique is based on the generation of an extended disjunctive program  $LP$  derived from the set of integrity constraints. The repairs for the database can be generated from the stable models of  $LP$ , whereas the computation of the consistent answers of a query  $(g, P)$  can be derived by considering the stable models of the program  $P \cup LP$  over the database  $D$ .

Let  $c$  be a universally quantified constraint of the form:

$$\forall X [ B_1 \wedge \dots \wedge B_k \wedge \text{not } B_{k+1} \wedge \dots \wedge \text{not } B_n \wedge \phi \supset B_0 ]$$

then,  $dj(c)$  denotes the extended disjunctive rule:

$$\begin{aligned} & \neg B'_1 \vee \dots \vee \neg B'_k \vee B'_{k+1} \vee \dots \vee B'_n \vee B'_0 \leftarrow (B_1 \vee B'_1), \\ & \dots, (B_k \vee B'_k), \\ & \quad (\text{not } B_{k+1} \vee \neg B'_{k+1}), \dots, (\text{not } B_n \vee \\ & \neg B'_n), \phi, (\text{not } B_0 \vee \neg B'_0), \end{aligned}$$

where  $B'_i$  denotes the atom derived from  $B_i$  by replacing the predicate symbol  $p$  with the new symbol  $p_d$  if  $B_i$  is a base atom, otherwise equal to false. Let  $IC$  be a set of universally quantified integrity constraints, then  $DP(IC) = \{ dj(c) / c \in IC \}$  whereas  $LP(IC)$  is the set of standard disjunctive rules derived from  $DP(IC)$  by rewriting the body disjunctions.

Clearly, given a database  $D$  and a set of constraints  $IC$ ,  $LP(IC)_D$  denotes the program derived from the union of the rules  $LP(IC)$  with the facts in  $D$  whereas  $SM(LP(IC)_D)$  denotes the set of stable models of  $LP(IC)_D$ , and every stable model is consistent since it cannot contain two atoms of the form  $A$  and  $\neg A$ . The following example shows how constraints are rewritten.

**Example 5.** Consider the following integrity constraints:

$$\begin{aligned} & \forall X [ p(X) \wedge \text{not } s(X) \supset q(X) ] \\ & \forall X [ q(X) \supset r(X) ] \end{aligned}$$

and the database  $D$  containing the facts  $p(a)$ ,  $p(b)$ ,  $s(a)$ , and  $q(a)$ .

The derived generalized extended disjunctive program is defined as follows:

$$\begin{aligned} & \neg p_d(X) \vee s_d(X) \vee q_d(X) \leftarrow (p(X) \vee p_d(X)) \vee (\text{not } \\ & s(X) \vee \neg s_d(X)) \vee (\text{not } q(X) \vee \neg q_d(X)). \\ & \neg q_d(X) \vee r_d(X) \leftarrow (q(X) \vee q_d(X)) \vee (\text{not } r(X) \\ & \vee \neg r_d(X)). \end{aligned}$$

The above rules can now be rewritten in standard form. Let  $P$  be the corresponding extended disjunctive Datalog program. The computation of the program  $P_D$  gives the following stable models:

$$\begin{aligned} M_1 &= D \cup \{ \neg p_d(b), \neg q_d(a) \}, \\ M_2 &= D \cup \{ \neg p_d(b), r_d(a) \}, \\ M_3 &= D \cup \{ \neg q_d(a), s_d(b) \}, \\ M_4 &= D \cup \{ r_d(a), s_d(b) \}, \\ M_5 &= D \cup \{ q_d(b), \neg q_d(a), r_d(b) \} \text{ and} \\ M_6 &= D \cup \{ q_d(b), r_d(a), r_d(b) \}. \end{aligned}$$

A (generalized) extended disjunctive Datalog program can be simplified by eliminating from the body rules all literals whose predicate symbols are derived and do not appear in the head of any rule (these literals cannot be true). As mentioned before, the rewriting of constraints into disjunctive rules is useful for both (1) making the database consistent through the insertion and deletion of tuples and (2) computing consistent answers leaving the database inconsistent.

## CONCLUSION

In the integration of knowledge from multiple sources, two main steps are performed: the first in which the various relations are merged together and the second in which some tuples are removed (or inserted) from the resulting database in order to satisfy integrity constraints.

The database obtained from the merging of different sources could contain inconsistent data. In this article, we investigated the problem of querying and repairing inconsistent databases. In particular, we presented the different techniques for querying and repairing inconsistent databases (Agarwal et al., 1995; Arenas et al., 1999; Greco & Zumpano, 2000; Lin & Mendelzon, 1996).

## FUTURE TRENDS

As a future trend, an interesting topic consists in specifying preference criteria so that selecting the preferable repairs among a set of feasible ones, that is, those better conforming to the specified criteria. Preference criteria introduce desiderata on how to update the inconsistent database in order to make it consistent; thus, they can be considered as a set of desiderata which are satisfied *if possible* by a generic repair. Therefore, informally a preferred repair is a repair that better satisfies preferences.

## REFERENCES

Agrawal, S., Keller, A.M., Wiederhold, G., & Saraswat, K. (1995). Flexible relation: An approach for integrating data from multiple, possibly inconsistent databases. *Proceedings of the IEEE International Conference on Data Engineering* (pp. 495-504).

Arenas, M., Bertossi, L., & Chomicki, J. (1999). Consistent query answers in inconsistent databases. *Proceedings of the International Conference on Principles of Database Systems* (pp. 68-79).

Arenas, M., Bertossi, L., & Chomicki, J. (2000). Specifying and querying database repairs using logic programs with exceptions. *Proceedings of the International Conference on Flexible Query Answering* (pp. 27-41).

Baral, C., Kraus, S., Minker, J., & Subrahmanian, V.S. (1991). Combining knowledge bases consisting of first order theories. *Proceedings of the International Symposium on Methodologies for Intelligent Systems* (pp. 92-101).

Bry, F. (1997). Query answering in information systems with integrity constraints. *Proceedings of the IFIP WG 11.5 Working Conference on Integrity and Control in Information System*.

Dung, P.M. (1996). Integrating data from possibly inconsistent databases. *Proceedings of the International Conference on Cooperative Information Systems* (pp. 58-65).

Greco, S., & Zumpano, E. (2000). Querying inconsistent databases. *Proceedings of the International Conference on Logic Programming and Automated Reasoning* (pp. 308-325).

Greco, S., & Zumpano, E. (2000). Computing repairs for inconsistent databases. *Proceedings of the International Symposium on Cooperative Database Systems for Advanced Applications* (pp. 33-40).

Greco, G., Greco, S., & Zumpano, E. (2001). A logic programming approach to the integration, repairing and querying of inconsistent databases. *Proceedings of the International Conference on Logic Programming*.

Greco, G., Sirangelo, C., Trubitsyna, I., & Zumpano, E. (2003). Preferred repairs for inconsistent databases. *Proceedings of the International Conference on Database Engineering and Applications Symposium*.

Lin, J. (1996). A semantics for reasoning consistently in the presence of inconsistency. *Artificial Intelligence*, 86(1), 75-95.

Lin, J., & Mendelzon, A.O. (1996). Merging databases under constraints. *International Journal of Cooperative Information Systems*, 7(1), 55-76.

Lin, J., & Mendelzon, A.O. (1999). Knowledge base merging by majority. In R. Pareschi & B. Fronhofer (Eds.), *Dynamic worlds: From the frame problem to knowledge management*. Kluwer.

Ullman, J.K. (2000). Information integration using logical views. 239(2), 189-210.

Wiederhold, G. (1992). Mediators in the architecture of future information systems. *IEEE Computer*, 25(3), 38-49.

## KEY TERMS

**Consistent Answer:** A set of tuples, derived from the database, satisfying all integrity constraints.

**Consistent Database:** A database satisfying a set of integrity constraints.

**Database Repair:** Minimal set of insert and delete operations which makes the database consistent.

**Data Integration:** A process providing a uniform integrated access to multiple heterogeneous information sources.

**Disjunctive Datalog Program:** A set of rules of the form:

$$A_1 \vee \dots \vee A_k \leftarrow B_1, \dots, B_m, \text{not } B_{m+1}, \dots, \text{not } B_n, k+m+n>0$$

where  $A_p, \dots, A_k, B_p, \dots, B_n$  are atoms of the form  $p(t_p, \dots, t_h)$ ,  $p$  is a predicate symbol of arity  $h$  and the terms  $t_p, \dots, t_h$  are constants or variables.

**Inconsistent Database:** A database violating some integrity constraints.

**Integrity Constraints:** Set of constraints which must be satisfied by database instances.

# Repairing Inconsistent XML Data with Functional Dependencies

**Sergio Flesca**

*DEIS Università della Calabria, Italy*

**Fillippo Furfaro**

*DEIS Università della Calabria, Italy*

**Sergio Greco**

*DEIS Università della Calabria, Italy*

**Ester Zumpano**

*DEIS Università della Calabria, Italy*

## INTRODUCTION

The World Wide Web is of strategic importance as a global repository for information and a means of communicating and sharing knowledge. Its explosive growth has caused deep changes in all the aspects of human life, has been a driving force for the development of modern applications (e.g., Web portals, digital libraries, wrapper generators, etc.), and has greatly simplified the access to existing sources of information, ranging from traditional DBMS to semi-structured Web repositories. The adoption by the WWW consortium (W3C) of XML (eXtensible Markup Language) as the new standard for information exchange among Web applications has led researchers to investigate classical problems in the new environment of repositories containing large amounts of data in XML format.

Great attention has also been recently devoted to the introduction of integrity constraints and the definition of normal forms for XML (Arenas & Libkin, 2003, 2004; Fan & Libkin, 2002; Vincent & Liu, 2003). XML allows a simple form of constraints to describe references obtained through ID/IDREF, but it does not actually provide a general mechanism for expressing semantic constraints like those commonly used in relational databases. The need of enriching the semantics of XML is so deep as a large amount of XML data originates in object-oriented and relational databases, where different forms of integrity constraints are used to add semantics to the collected information.

This work stems from the need of enriching the semantics of XML documents. This need is attested by several new works which introduce different forms of constraints to XML documents (Arenas, Fan & Libkin, 2002, 2004; Buneman et al., 2001, 2002; Fan & Libkin, 2002; Fan & Simeon, 2000; Vincent et al., 2004; Yang, Yu & Wang,

2001). Most of them introduce a simple form of constraints such as keys and foreign keys, whereas some others attempt to extend the class of integrity constraints associated with XML documents.

Obviously, reasoning about constraints in the presence of an incomplete knowledge of the data structure is rather complex so that some of these attempts are likely to be a purely theoretical exercise. In fact, their practical applicability follows the solution of non-trivial problems such as the implication and interaction among constraints which are far from being solved. In the presence of constraints on data, an XML document may result in being inconsistent; that is, it does not respect some constraint. The following example shows the case of an inconsistent XML document.

**Example 1.** Consider the following XML document representing a book collection:

```
<bib>
  <book isbn="0-451-16194-7">
    <title> A First Course in Database Systems </
title>
    <author> Ullman </author>
    <author> Widom </author>
    <publisher> Prentice-Hall </publisher>
  </book>
  <book isbn="0-451-16194-7">
    <title> Principles of Database and Knowledge-
Base Systems
    </title>
    <author> Ullman </author>
    <publisher> Computer Science Press </pub-
lisher>
  </book>
</bib>
```



and the functional dependency  $\text{bib.book.@isbn} \rightarrow \text{bib.book.title.S}^1$ , stating that two books with the same isbn must have the same title.

The above document does not satisfy this functional dependency, as the first and the second book have the same isbn attribute but different titles.

The above example shows that, generally, the satisfaction of constraints cannot be guaranteed, thus, in the presence of an XML document which must satisfy a set of constraints, we have to manage potential inconsistencies of data. This problem has been recently investigated for relational databases and several techniques based on the computation of repairs (minimal sets of insert/delete operations), and consistent answers have been proposed in this context (Arenas, Bertossi & Chomicki, 1999; Greco & Zumpano, 2000). However, these techniques cannot easily be extended to XML data because of the different structure of data and the different nature of constraints.

The document of the previous example can be repaired by performing one of the following minimal sets of update operations:

- replace the string “*A First Course in Database Systems*” with the title “*Principles of Database and Knowledge-Base Systems*”;
- replace the string “*Principles of Database and Knowledge-Base Systems*” with the title “*A First Course in Database Systems*”;
- assign a new, different value to one of the two isbn attributes so that there are no two books with the same isbn.

This means that the violation of a functional dependency could be repaired by applying several sets of possible update operations, yielding a consistent scenario of the information. In our framework, we prefer the repairs performing minimal sets of changes to the original document, in the same way as well known approaches proposed for relational database repairing.

We also address the problem of extracting “reliable” information from inconsistent documents. To this end, we define two different semantics for queries, which are evaluated on the repaired version of the given document, instead of the (inconsistent) original one. A wider discussion of the notions and contributions introduced in this article is provided in Flesca et al. (2003).

## BACKGROUND

A functional dependency  $A \rightarrow B$  in a relational database  $D$  models the correspondence between  $A$  and  $B$  values in the tuples of  $D$ . However, there is no standard concept for tuple in the XML context. In this section, we recall the

notion of functional dependency in the XML setting proposed in Arenas and Libkin (2004) and Arenas et al. (2002)<sup>2</sup>, which will be used in the following as a basis for our framework. Before introducing functional dependencies for XML, we present the tree-based representation model which will be adopted in the rest of this work, and then we provide the concept of *tree tuple*, corresponding to the concept of tuple in relational databases.

From now on, XML documents will be represented by means of labeled trees (called “XML trees”) whose nodes correspond to either elements, attributes, or string values. In particular, nodes corresponding to elements have a single label (representing the tag name), whereas nodes corresponding to attributes have two labels, representing the attribute name and value, respectively. The text content of an element will be represented by a distinguished node (marked with the symbol “S”) labeled with a string equal to the text content of the element. An example of XML tree is shown in Figure 1.

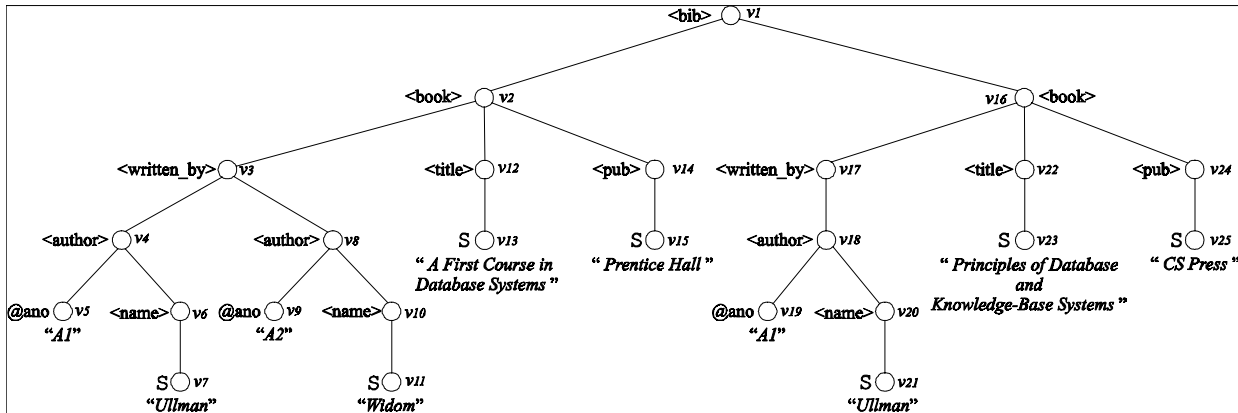
The information represented in an XML document can be extracted by means of path expressions identifying nodes of the corresponding XML tree. A path expression is a sequence of symbols (i.e., tag names, attribute names, or the symbol “S”) occurring in the XML tree, identifying traversals over it. In more detail, the result of a path expression ending with an element name is the set of node identifiers which can be reached, starting from the root, walking through a sequence of nodes whose labels satisfy the given expression. Otherwise, if the path expression ends with either an attribute name or the symbol “S”, the result is a set of strings representing the attribute values (or the text content of elements) which can be reached by means of path satisfying the given expression. For instance, the path expression  $\text{bib.book.title}$  applied on the XML tree of Figure 1 returns the set  $\{v12, v22\}$ , whereas  $\text{bib.book.written\_by.author.name.S}$  returns the set  $\{\text{“Ullman”}, \text{“Widom”}\}$ .

Informally, a tree tuple groups together nodes of the document which are semantically correlated, according to the structure of the tree. For instance, a tree tuple of the XML tree  $XT$  of Figure 1 consists of a sub-tree which contains information about a book. Observe that each book is possibly described by more than one tree tuple, as each tree tuple contains the information of only one author.

**Definition 1.** Given an XML tree  $XT$ , a tree tuple  $t$  of  $XT$  is a maximal sub-tree of  $XT$  such that, for every path expression  $p$  defined on  $XT$ ,  $t.p$  contains at most one element.

**Example 2.** Consider the XML tree  $XT$  of Figure 1. The sub-tree of  $XT$  shown in Figure 2(a) is a tree-tuple, whereas the sub-tree in Figure 2(b) is not a tree tuple (it contains

Figure 1. An XML tree



two authors of the same book). This means that each book stored in  $XT$  can correspond to more than one tree tuple: each tree tuple corresponds to one of the book authors. Note that any sub-tree of the tree in Figure 2(a) is not a tree tuple, as it would not be maximal.

**Definition 2.** Given an XML tree  $XT$ , a functional dependency on  $XT$  is an expression of the form  $X \rightarrow p$ , where  $X$  is a finite nonempty set of path expressions defined on  $XT$ , and  $p$  is a single path expression on  $XT$ .

Given an XML tree  $XT$  and a functional dependency  $F: X \rightarrow p$ , we say that  $XT$  satisfies  $F$  (namely  $XT \models F$ ) if for each pair of tree tuples  $t_1, t_2$  of  $XT$ ,  $t_1.X = t_2.X \wedge t_1.X \neq \emptyset \Rightarrow t_1.p = t_2.p$ . Given a set of functional dependencies  $FD = \{F_1, \dots, F_n\}$  over  $XT$ , we say that  $XT$  satisfies  $FD$  if it satisfies  $F_i$  for every  $i \in 1..n$ .

**Example 3.** Consider the XML tree  $XT$  of Figure 1. The constraint that the attribute @ano identifies univocally the (value of the) name of every author can be expressed with the following functional dependency:

$$\text{bib.book.written\_by.author.@ano} \rightarrow \text{bib.book.written\_by.author.name.S}$$

To say that two distinct authors of the same book cannot have the same value of the attribute ano, we can use the following FD:

$$\{\text{bib.book}, \text{bib.book.written\_by.author.@ano}\} \rightarrow \text{bib.book.written\_by.author}$$

## REPAIRING INCONSISTENT XML DATABASES

In this section, we present an approach to the problem of repairing XML documents which are inconsistent w.r.t. a given set of functional dependencies. A possibly inconsistent XML document can be repaired by taking two different kind of actions:

1. by changing the value of an attribute or the content of an element;

Figure 2. Two sub-trees of the XML tree of Figure 1

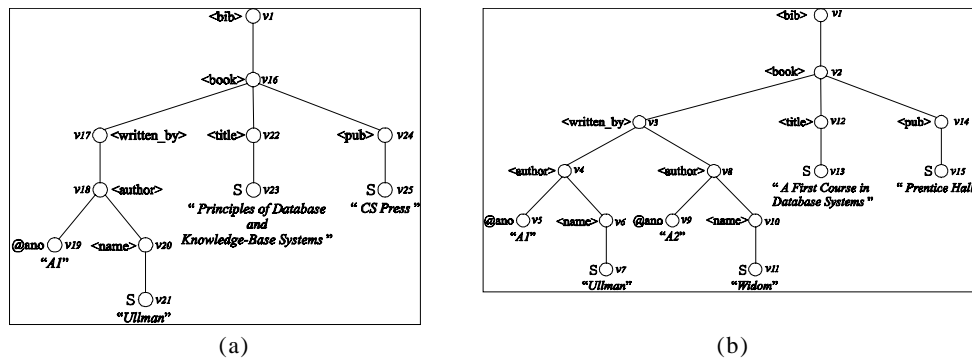
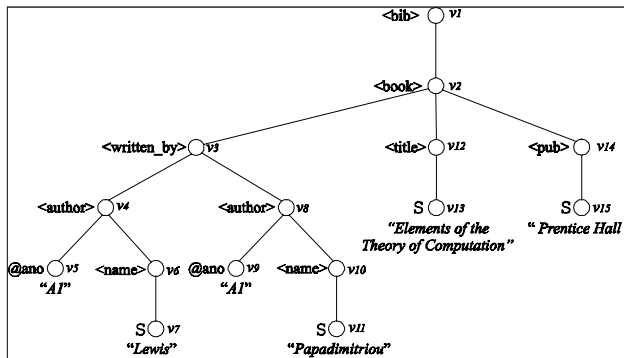


Figure 3. An XML tree



- by marking some of the attributes or elements of the document as “unreliable”.

**Example 4.** Consider the XML tree  $XT$  of Figure 3, and suppose that we are given the following functional dependency:

$$\{ \text{bib.book}, \text{bib.book.written\_by.author.@ano} \} \rightarrow \text{bib.book.written\_by.author}$$

$XT$  does not satisfy the above FD, as the two author elements contained in the same book, have the same value of the attribute `ano`, whereas the above FD requires that, for each book, there is only one author having a given `ano` value.

The constraint in the above example may not be satisfied for two possible reasons: (1) one of the two values is incorrect; (2) one of the two author elements is incorrect. Therefore, two repairing strategies are possible. If we assume that the former of the two errors occurs, we are induced to change the `ano` value of one of the authors. That is, we can make  $XT$  consistent w.r.t. the given FD by assigning a new value (i.e., a skolem constant, denoted as ‘ $\perp_1$ ’) to the attribute `ano` of any of the author elements, see Figure 4(a). Otherwise, if we assume that the latter error occurs (i.e., one of the two author elements is incorrect), we choose to mark one of the two authors having the same `ano` as “unreliable”, see Figure 4(b), where unreliable nodes are marked with the symbol “\*”. Marking a node as unreliable means that it contains some information of which trustworthiness is not guaranteed. This implies that when checking a functional dependency on a repaired document, nodes marked as unreliable are not considered. That is, marking nodes as unreliable makes the XML tree consistent w.r.t. the given functional dependency in a *weaker* sense: if an XML tree  $XT$  does not satisfy the functional dependency  $F$ :

$\{x_1, \dots, x_n\} \rightarrow p$ , we say that  $XT$  *weakly satisfies*  $F$  if either  $t_{1,x_i}$  or  $t_{2,x_i}$  is marked as unreliable (for some  $i \in [1..n]$ ), or either  $t_{1,p}$  or  $t_{2,p}$  is marked as unreliable.

However, in the case of the XML tree of Figure 3, the latter strategy (marking an author element as unreliable) changes a larger portion of the document, since all the sub-elements of author are marked as unreliable, whereas the first strategy only changes its `ano`. Repair strategies performing smaller changes to the original document will be preferred, in the same way as in well-known approaches in the relational database scenario (Arenas et al., 1999; Greco & Zumpano, 2000). Observe that we prefer marking a node as unreliable rather than deleting it, since removing elements from an XML document can lead to undesired side effects: for instance, deleting a node can yield to a new document not conforming the DTD associated to the original one.

When evaluating a query (expressed by means of a path expression) on a repaired document, nodes marked as unreliable are not considered.

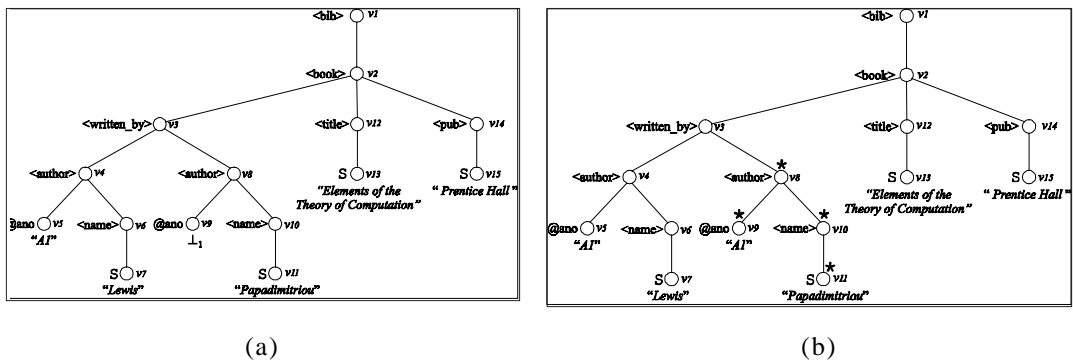
Our repairing strategy can be summarized as follows. Given a document  $XT$  and a functional dependency such that two tuples  $t_1, t_2$  of  $XT$  do not satisfy  $X \rightarrow y$  [i.e.  $t_1.X = t_2.X$  but  $t_1.y \neq t_2.y$ ], we can repair  $XT$  in two ways:

- change the value of any  $x \in X$  in one of the two tuples in order to make (if possible)  $t_1.X \neq t_2.X$ . Two cases may occur:
  - if  $t_1.x$  and  $t_2.x$  are nodes, we cannot make a node different to itself; then, we mark  $t_1.x$  as *unreliable*;
  - if  $t_1.x$  and  $t_2.x$  are strings (attribute or element values), we assign a new different value (skolem constant, namely ‘ $\perp$ ’) to either  $t_1.x$  or  $t_2.x$ ;
- change the value of  $y$  in one of the two tuples, in order to make (if possible)  $t_1.y = t_2.y$ . Two cases may occur:
  - if  $t_1.y$  and  $t_2.y$  are nodes, we cannot merge two distinct nodes into a unique one; therefore, we mark either  $t_1.y$  or  $t_2.y$  as *unreliable*;
  - if  $t_1.y$  and  $t_2.y$  are strings (attribute or element values), we assign the value of  $t_1.y$  to  $t_2.y$ , or vice versa.

Finally, among all possible repairs, we consider minimal ones.

Indeed, for a given document, more than one repair could be minimal. Therefore, when evaluating a query  $q$  (expressed by means of a path expression) on an inconsistent document, we consider two possible semantics:

Figure 4. Two possible repairs of the XML tree of Figure 3



1. **Possible answer:** Set of nodes (or values) which belong to the answer of  $q$  returned on at least one repaired document; and
2. **Certain answer:** Set of nodes (or values) which belong to every answer of  $q$  returned on every possible repaired document.

Observe that the answer of a query on a repaired document does not comprise attribute and element values which are marked as unreliable.

**Example 5.** Consider the XML tree of Figure 3 and the functional dependency `bib.book.written_by.author.@ano → bib.book.written_by.author.name.S`, stating that two authors with the same `ano` must have the same name. For the path query `bib.book.title.S`, both the possible and certain answers consist in the set {"Elements of the Theory of Computation"}. In fact, all possible repairs of the given XML tree do not change the element title.

As regards the path query `bib.book.author.name.S`, the possible answer is the set {"Lewis", "Papadimitriou"}, whereas the certain answer is the empty set. In fact, four minimal repairs can be performed. Two of these repairs consist in replacing one of the two author names with the new value `^`, whereas the other two repairs consist in replacing the string "Papadimitriou" with "Lewis", and, respectively, the string "Lewis" with "Papadimitriou". Therefore, there exists at least one repair yielding to a document where the string "Lewis" does not occur and at least one repaired document which does not contain the string "Papadimitriou".

### FUTURE TRENDS

We are currently investigating how to extend our repairing strategy when further constraints (such as DTD and inclusion dependencies) are defined.

### CONCLUSION

We have defined a framework for repairing XML documents which are inconsistent w.r.t. to a given set of functional dependencies. Repairs consist in minimal sets of update operations which either change attribute and element values or mark elements as unreliable. We have also addressed the problem of extracting reliable information from inconsistent documents after applying our repairing strategy.

### REFERENCES

Arenas, M., Bertossi, L., & Chomicki, J. (1999, Month 00). Consistent query answers in inconsistent databases. *Proceedings of the Symposium on Principles of Database Systems (PODS)*, Philadelphia.

Arenas, M., Fan, W., & Libkin, L. (2002). On verifying consistency of XML specifications. *Proceedings of the Symposium on Principles of Database Systems (PODS)*, Madison, Wisconsin.

Arenas, M., Fan, W., & Libkin, L. (2004). Consistency of XML specifications. In L. Bertossi, A. Hunter, & T. Schaub (Eds.), *Inconsistency tolerance*. Springer-Verlag.

Arenas, M., & Libkin, L. (2003). An information-theoretic approach to normal forms for relational and XML data.

*Proceedings of the 22nd Symposium on Principles of Database Systems (PODS)*, New York.

Arenas, M., & Libkin, L. (2004). A normal form for XML documents. *ACM Transactions on Database Systems*, 29(1).

Buneman, P., Davidson, S.B., Fan, W., Hara, C.S., & Tan, W.C. (2001). Reasoning about keys for XML. Revised Papers from the *8th International Workshop on Database Programming Languages (DBPL)*, Frascati, Italy.

Buneman, P., Davidson, S.B., Fan, W., Hara, C.S., & Tan, W.C. (2002). Keys for XML. *Computer Networks*, 39(5).

Fan, W., & Libkin, L. (2002). On XML integrity constraints in the presence of DTDs. *Journal of the ACM*, 49(3).

Fan, W., & Simeon, J. (2000). *Integrity constraints for XML. Proceedings of the 19th ACM Symposium on Principles of Database Systems (PODS)*, New York.

Flesca, S., Furfaro, F., Greco, S., & Zumpano, E. (2000). Repairs and consistent answers for XML data with functional dependencies. *Proceedings of the XML Database Symposium (XSym)*, Berlin, Germany.

Greco, S., & Zumpano E. (2000). Querying inconsistent databases. *Proceedings of the 7th International Conference on Logic for Programming and Automated Reasoning (LPAR)*, Reunion Island, France.

Liu, J., Vincent, M.W., & Liu, C. (2003). Functional dependencies, from relational to XML. *Proceedings of the 5th Ershov Memorial Conference Perspectives of System Informatics (PSI)*, Novosibirsk, Russia.

Vincent, M.W., Schrefl, M., Liu, J., Liu, C., & Dogen, S. (2004). Generalized inclusion dependencies in XML. *Proceedings of the 6th Asia Pacific Web Conference (APWeb)*, Hangzhou, China.

Vincent, M.W., & Liu, J. (2003). Functional dependencies for XML. *Proceedings of the 5th Asia Pacific Web Conference (APWeb)*, Xian, China.

Yang, X., Yu, G., & Wang, G. (2001). Efficiently mapping integrity constraints from relational database to XML document. *Proceedings of the 5th East European Conference on Advances in Databases and Information Systems (ADBIS)*, Vilnius, Lithuania.

## KEY TERMS

**Consistent Answer:** Data satisfying both the query and all integrity constraints defined on the given database.

**Consistent Database:** A database satisfying a set of integrity constraints.

**Data Integration:** A process providing a uniform integrated access to multiple heterogeneous information sources.

**Database Repair:** Minimal set of insert and delete operations which makes the database consistent.

**Inconsistent Database:** A database violating some integrity constraint.

**Integrity Constraints:** Set of constraints which must be satisfied by database instances.

## ENDNOTES

<sup>1</sup> The symbol “S” is used to extract the text content from an element.

<sup>2</sup> An alternative definition has been proposed in Vincent and Liu (2003) and Liu, Vincent, and Liu (2003).



# Replication Mechanisms Over a Set of Distributed UDDI Registries

**Zakaria Maamar**

*Zayed University, UAE*

## INTRODUCTION

This paper presents a research initiative, which aims at developing replication mechanisms for the dynamic management of the content of several Universal Description, Discovery, and Integration (UDDI) registries (Curbera et al., 2003). These replication mechanisms are intended to be deployed in an environment of Web services (Papazoglou & Georgakopoulos, 2003). By content of an UDDI registry, we mean the announcements of Web services that providers post on the UDDI registry. Unlike other research initiatives in the field of Web services that essentially consider a single UDDI registry and assume a wired and stable communication infrastructure, the following aspects constitute the core of this research initiative:

- Several UDDI registries are spread across different regions. An UDDI registry is aware of the presence of other peers but does not perform any direct exchange of information on its content with them. The UDDI registries may belong to different businesses, have different usage policies, and pose various requirements on acceptable announcements and retrieval demands of Web services.
- There is no predefined communication infrastructure between the distributed UDDI registries. An infrastructure of type wired or wireless for direct interactions can be set up after assessing the importance of the exchange between the UDDI registries. In addition, an UDDI registry may be called to disappear if its owner decides to withdraw it.
- Absence of a centralized component that manages and coordinates the UDDI registries. It is noted that a central authority has always constituted a bottleneck in a system operation (Penserini et al., 2003). On the one hand, each UDDI registry is independent in defining the announcements of providers that it accepts and the retrieval demands of users that it satisfies. The definition of what to accept and what to satisfy is based on a set of UDDI registry-defined policies. On the other hand, each provider is independent in selecting the UDDI registries to which it will post its announcements

of Web services. The selection of where to post is based on a set of provider-defined policies.

In a Web services scenario, an UDDI registry participates in two operations. The first operation consists of receiving the announcements of the description of Web services (also called services) from providers. After posting the announcements, the second operation consists of searching the registry content for the services that satisfy specific needs upon users' needs. Examples of needs are multiple, varying from hotel booking and car rental to weather forecasts. The search consists of identifying the relevant services and indicating who offers them so that the identified services can be triggered after a potential composition (Casati et al., 2003). It is accepted that the advantages of Web services are highlighted by their capacity to be composed into high-level business processes referred to as composite services (Berardi et al., 2003). However, since the announcements of services are submitted to multiple UDDI registries, this results in a different content across the registries.

Targeting the dynamic management of multiple UDDI registries has some overlapping with the well-known problem of information total-replication over a set of distributed databases. An immediate solution to the UDDI-registry dynamic management is to flood the communication infrastructure with the new content of any UDDI registry that has been subject to changes. Changes in UDDI registries are expected to become frequent as the number of Web services of providers continues to grow. While the flooding seems to be a suitable solution for the context of a wired communication infrastructure, the lack of a reliable and permanent communication infrastructure is a major obstacle to this solution deployment. It was observed that the traditional database approaches for collecting, caching, and indexing data of interest in monolithic contexts become obsolete in global computing contexts (Karakasidis & Pitoura, 2002). In addition, unlike the case of a wired environment, the assumption of large bandwidth availability, low error rates, and always-on connectivity are invalidated in a wireless environment. Therefore, another alternative is required for the dynamic manage-

ment of the content of UDDI registries. It will be outlined throughout this article how mobile users constitute the vehicles of supporting the content exchange between the UDDI registries (i.e., content replication). It is important to strengthen that this support is done in a transparent way to users because of the use of software agents, which act on their behalf (Jennings, Sycara & Wooldridge, 1998).

Each UDDI registry is associated with a structure known as *cluster* of Web services. Several clusters exist across the communication infrastructure so that providers can connect to the most appropriate cluster according to various criteria such as proximity to and workload status of a cluster. The connection between providers and clusters is of type wired. For tracking purposes, a provider cannot be connected to more than one cluster, which means that a provider cannot post its announcements of Web services on the UDDI registries of multiple clusters. The cluster in which a provider declares its services for the first time is called *master*. Interesting is the situation where providers have similar Web services but respectively announce their Web services in separate UDDI registries. In Benatallah, Sheng, and Dumas (2003), service similarity is explained with the concept of service communities where alliances are formed among a potentially large number of services performing the same operation types. Unless some appropriate exchange mechanisms are made available, an UDDI registry would never be aware of the existence of similar services in other registry peers. Besides that, for a users wishing to satisfy their needs by triggering or composing Web services, users should be given the opportunity to consider all the existing services regardless of where they are announced. The two aforementioned scenarios (i.e., service similarity and users' needs) shed light on the importance of supporting a content exchange between the UDDI registries. This content exchange requires deploying appropriate replication mechanisms.

A part of the solution of the dynamic management problem of UDDI registries relies on users who, first, are mobile and, second, have mobile devices (e.g., cell phones, personal digital assistants). The other part of the solution relies on software agents. It is accepted that software agents are suitable candidate for performing the composition operations of Web services on behalf of users (Huhns, 2002; Kuno & Sahai, 2002). The solution, which is outlined in this article, combines users and software agents to constitute what we refer to as messengers. Briefly about the operation of messengers, a software agent resides in the mobile device of a user. The agent caches a description of the list of Web services that were involved in the satisfaction of a user's needs. On behalf of providers, users post services on

various UDDI registries that are associated with clusters known as *slaves*. Because users have mobile devices, mobile support stations manage these devices when it comes to identifying their physical location and handling their incoming and outgoing messages/calls (Maamar, Ben-Younes & Al-Khatib, 2003). A mobile support station communicates with mobile users within its radio coverage area known as *wireless cell*. For the needs of this initiative, each cluster of Web services is attached to a mobile support station. Therefore, when a user enters a new cell (i.e., the user becomes under the coverage area of a new mobile support station), an exchange of information between the software agent of the user and the UDDI registry is conducted. This exchange enables updating this registry content. It is deemed appropriate to mention that users do not have to visit all the clusters. Their association with a mobile support station depends on their route to various places such as work, gym, and so forth.

Because a UDDI registry receives information on Web services from two independent sources, namely, providers of Web services and agents of users, the services are decomposed into two types: internal and external. Internal services are announced in an UDDI registry of a master cluster (the providers take care of the announcements). This registry has full control over the internal services by guaranteeing, for example, their QoS arguments. External services are always announced in an UDDI registry of a slave cluster (the agents of users take care of the announcements). This registry cannot, for example, guarantee the QoS arguments of the external services and their availability in their respective provider hosts for triggering purposes. Handling the features of external services constitutes one of the challenges of managing the content of several UDDI registries.

In this initiative, the exchange of the content of the UDDI registries does not target a total replication. Instead, a partial replication, which evolves over time to reach the level of a total replication, is aimed. A total replication between the UDDI registries might happen subject to the following factors:

- **The route of users:** users are not forced to visit all the clusters so that the UDDI registries are fed with new content. The involvement of users in the dynamic management of the UDDI registries does not have to be a burden on them. Because of the diversity of the routes of users, the update of an UDDI registry occurs each time these users are linked to a new support mobile station of a cluster, thus, in the vicinity of a new UDDI registry.
- **The different policies that exist such as provider-defined policies:** for instance, each regis-

try is independent in defining the announcements of providers that it accepts and the retrieval demands of users that it satisfies. In addition, each provider is independent in selecting the UDDI registries to which it submits its announcements of Web services.

In the next section, the rationale of using software agents in the management of the content of several UDDI registries is outlined. Afterwards, some of the related works are discussed. The different types of agents, which are responsible for deploying this management, are discussed. Finally, a conclusion is presented.

## BACKGROUND

Besides the availability of several approaches and technologies related to the deployment of Web services (e.g., SOAP, UDDI, Salutation), they are all tailored to a wired context. In this context, the computing resources are fixed and connected through a permanent and reliable communication infrastructure. The application of these approaches and technologies to a wireless context is not straightforward. Indeed, major adjustments are required because of multiple obstacles such as potential disconnections of mobile devices, unrestricted mobility of persons, and power scarcity of mobile devices. These obstacles highlight the suitability of software agents as potential candidates to overcome them. First a software agent is autonomous. Thus, it can make decisions on the users' behalf while they are disconnected (i.e., off-line). Second, a software agent can be mobile. Therefore, it can move from one host to another. Continuous network connectivity is not required. Third, a software agent is collaborative. Therefore, it can work with other agents that identify, for example, the providers of Web services. Last but not least, a software agent is reactive. Consequently, it can monitor the events that occur in the user's environment so that relevant actions can be promptly taken.

Scenarios where people while on the move electronically interact with their surrounding environment have been reported in Jose, Moreira, and Rodrigues (2003). These scenarios back our approach of getting users transparently engaged in various operations such as updating UDDI registries. Our initiative is at the crossing point of several initiatives on Web services, UDDI registry, and wireless. While these concepts are being independently studied (except for Web services and UDDI), we aim at their combination in the same framework. In addition, it is mentioned that the new version of the UDDI specification supports the fact that several registries might be

present at the same time, thus, have to interact. With this kind of specification, various challenges are faced when dealing with hundreds of registries (if not thousands) during service publication and discovery.

In Valavanis et al. (2003), MobiShare project provides a middleware system for offering ubiquitous connectivity to mobile devices. A user's mobile device is seen as a source of services, a requestor of services, or both. Service in MobiShare means the data that devices decide to publicly make available. Data availability depends on the status (on/off) and location (dependent on the coverage area) of a device. In DBGlobe project, the aim is to develop data and metadata management techniques in a global computing context (Karakasidis & Pitoura, 2002). DBGlobe considers mobile entities as primary data stores and broadens the data management focus to address various issues such as mobility, autonomy, and scalability. To make data widely available, DBGlobe relies on chained hierarchies of directories. In case of an unsatisfied request at the level of a directory, the request is forwarded to a higher geographical authority. In the eNcentive project (Ratsimor et al., 2003), peer-to-peer electronic marketing in mobile ad-hoc environments is studied. The eNcentive project employs an intelligent marketing scheme by allowing users to collect information like sales promotions and discounts. The marketing scheme relies on users who propagate promotions and discounts to other users with the same interests and preferences in the network. Users participating in eNcentive take advantage of circulating announcements since businesses that originally issued the announcements reward the active distributors with additional promotions and compensations.

The environment of Web services that is targeted in this article has some similarities with peer-to-peer configurations. There is no centralized component that manages and coordinates the UDDI registries. In addition, all UDDI registries have equivalent capabilities and responsibilities in terms of accepting the announcements of Web services and satisfying the retrieval requests of Web services. Because the communication infrastructure between the distributed UDDI registries is not pre-defined, the flooding solution was abandoned, and the use of messengers was promoted. The flooding practice should not be adopted in peer-to-peer environments. However, Castano et al. (2003) observe that the solutions proposed for content retrieval in these environments often exploit either flooding or broadcasting to disseminate the queries when the location of a searched content is unknown.

## AGENTS FOR UDDI REGISTRIES

The agentification of the dynamic management of the UDDI registries results in the identification of three types of software agents: provider-agent, UDDI-agent, and user-agent. Figure 1 illustrates the agents according to the clusters of Web services, the UDDI registries, and the mobile support stations. Usually, the coverage areas (represented by circles in Figure 1) of the mobile support stations overlap. However, to keep the figure clear, the overlapping is not represented. We recall that there is no predefined communication infrastructure between the clusters of Web services. Figure 1 also depicts the notion of messenger, which is the couple (user, user-agent) while on the move. Users with their mobile devices are always under the management of a particular mobile support station. When a user moves to a different place, which is outside the coverage area of a mobile support station, a handover occurs between this station and the new mobile support station, which is responsible for covering this place. The messenger-based operation for the dynamic management of the content of several UDDI registries is decomposed into three stages. The first stage consists of storing the description of the Web services that satisfy the needs of users. The second stage, which is the core of this research initiative, consists of updating the UDDI registries using messengers. Finally, the third stage consists of identifying the Web services that satisfy the needs of users considering that several UDDI registries with different contents are deployed.

A provider-agent identifies a provider (i.e., a business) that posts its Web services on the UDDI registries of the multiple clusters. However, a provider is only authorized to connect to one master cluster so that it can use its UDDI registry. The services announced in the UDDI registry of a master cluster are labeled as internal. This labeling helps in (1) identifying the UDDI registry

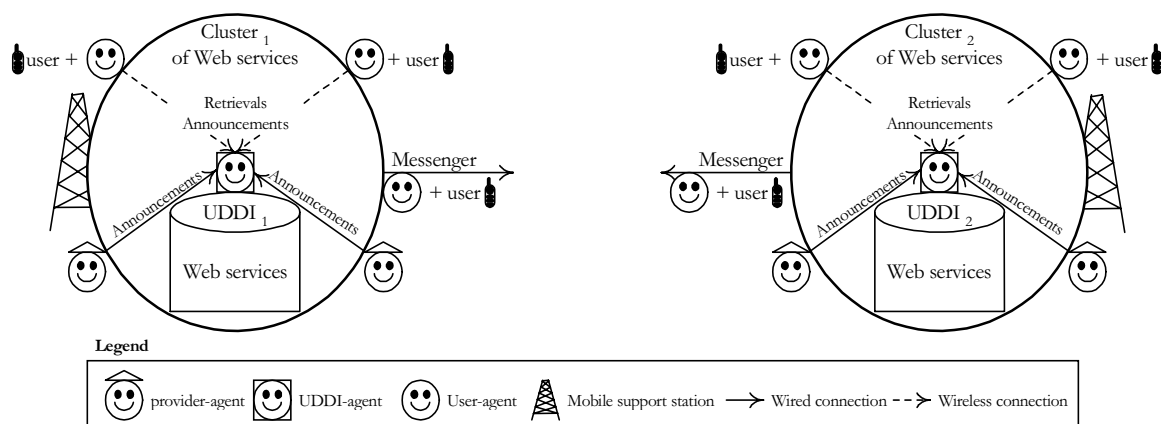
where the services are announced for the first time; (2) knowing the location of the providers to whom the services belong; (3) denying all the operations of user-agents that aim at updating the description of the services; and (4) naming the UDDI-agent that ensures that the parameters of the services (e.g., execution cost, execution time, QoS) are satisfied during execution.

Since a provider only announces its Web services in one UDDI registry, messengers take care of the UDDI registries of the remaining slave clusters. For announcement purposes in the slave clusters, the messengers adopt certain policies. The services posted on the UDDI registry of a slave cluster are labeled as external. This labeling helps (1) indicate that a certain messenger has announced the services, (2) inform that the services can always be subject to unpredicted changes, and (3) state that the UDDI-agent cannot guarantee the service execution parameters.

Because users are heavily engaged in the announcement of external Web services, the agreement of the respective providers of these Web services is required. For various reasons such as privacy (e.g., a provider does not want to announce all its services in a certain UDDI registry) and security (e.g., a provider is afraid that its announcements of services will be altered in a certain UDDI registry), a provider states in the description of a Web service that is submitted to an UDDI registry of a master cluster if this service can be announced in the UDDI registries of the slave clusters.

A user-agent resides in the mobile device of a user constituting a messenger when they move. The main duty of a user-agent is to satisfy its user's needs (e.g., car rental, hotel booking) after it identifies and selects the Web services with the help of an UDDI registry. To this purpose, the user-agent initiates interactions with the UDDI-agent of an UDDI registry. The consideration of a specific UDDI registry depends on the current location of

Figure 1. Agentification of distributed UDDI registries





the user with regard to the mobile support station that manages the mobile device. Once the services are identified and triggered for execution, the user-agent caches in the user's mobile device various information such as the identifier of the services, the UDDI registry with whom the user-agent has dealt with to obtain the services, and the providers of services and their location according to the master clusters. This information can be potentially announced to distinct UDDI registries of slave clusters. This is conducted after verifying the authorization policies of the announcements.

Users are always attached to one cluster because of the mobile support station. The type of cluster (master or slave) is not relevant. When a user moves to a different place, which is outside the coverage of the current support station, the user's mobile device becomes under the management of a new mobile support station. This means that the user-agent can now deal with the UDDI registry of the new cluster of Web services. The user-agent keeps track of all the clusters that it has visited in the past, its last date of visit, and the kind of information it submitted previously to their respective UDDI registry. If the user-agent notes that the information it caches is valuable to the UDDI-agent (what it was submitted vs. what it can now be submitted), a wireless communication is established between the two agents on a request basis from the user-agent. The rationale of the communication is to transfer the information on the Web services to the UDDI registry so that its content can get updated. The transfer obeys three categories of policies: provider-defined, UDDI registry-defined, and user-defined.

An UDDI-agent is deployed on top of an UDDI registry. The UDDI-agent interacts through wired connections with provider-agents for their announcements of services of type internal. In addition, the UDDI-agent interacts through wireless connections with user-agents for their retrieval requests of services and announcements of services of type external.

### **FUTURE TRENDS**

One of the trends in the dynamic management of UDDI registries consists of integrating ad-hoc networking into the operation of messengers. This is motivated by the fact that messengers can constitute the components of this type of network, besides the compliance of messengers with the definition of what an ad-hoc network is. As a prerequisite, the messengers have to get the authorizations to set up such a network from their respective users and are in the vicinity of each other. Because

routes of users and user-defined policies constrain the messengers in their information exchange on service descriptions with UDDI registries, this exchange can be boosted by allowing messengers to collect further information from their peers. It is noted that the current configuration in Figure 1 illustrates a messenger conveying information related to one user. The future configuration, which relies on ad-hoc networking, enables messengers to convey information related to multiple users. For security reasons, messengers do not accept anything from their peers that can be executed on the mobile devices.

Another future trend consists of enhancing messengers with contextual information (Brezillon, 2003). This is motivated by the fact that different factors can contextualize the information exchange that will occur between messengers and UDDI registries. Several factors can be cited, among them, location of exchange, period of exchange, and type of policy violation. Because of the contextual information that can embed UDDI registries and messengers, several types of questions can be handled such as which messengers are attached to which UDDI registry? What services is a messenger submitting to a UDDI registry? And, what routes has a messenger passed by?

### **CONCLUSION**

In this article, we outlined our approach for ensuring the replication across a set of distributed UDDI registries. The approach relies on the fact that users are mobile and the latest developments happening in the field of mobile devices (more storage capacity, more computing resources, and more advanced features). To manage the content of the UDDI registries, different types of policies were put forward stating, for example, where to announce, what to announce, and what to accept. These policies allow considering several aspects in the announcement of Web services such as security, privacy, and trustworthiness. Because of the lack of a predefined communication infrastructure that connect the UDDI registries, a flooding-based solution was discarded. Acting as messengers, users and their software agents support the exchange of content between the UDDI registries.

### **ACKNOWLEDGMENTS**

The author acknowledges the contributions of H. Yahyaoui and Q. Mahmoud to the research initiative presented in this paper.



## REFERENCES

- Benatallah, B., Sheng, Q.Z., & Dumas, M. (2003). The self-serv environment for Web services composition. *IEEE Internet Computing*, 7(1), 62-66.
- Berardi, D., Calvanese, D., De Giacomo, G., Lenzerini, M., & Mecella, M. (2003). A foundational vision for e-services. *Proceedings of The Workshop on Web Services, e-Business, and the Semantic Web (WES'2003) in conjunction with the 15th International Conference on Advanced Information Systems Engineering (CAiSE'2003)*, Klagenfurt/Velden, Austria.
- Brezillon, P. (2003). Focusing on context in human-centered computing. *IEEE Intelligent Systems*, 18(3), 29-34.
- Casati, F., Shan, E., Dayal, U., & Shan, M.C. (2003). Business-oriented management of Web services. *Communications of the ACM*, 46(10).
- Castano, A., Ferrara, S., Montanelli, S., Pagani, E., & Rossi, G.P. (2003). Ontology-addressable contents in P2P networks. *Proceedings of the 1st Workshop on Semantics in Peer-to-Peer and Grid Computing in conjunction with the 12th International World Wide Web Conference (WWW'2003)*, Budapest, Hungary.
- Curbera, F., Khalaf, R., Mukhi, N., Tai, S., & Weerawarana, S. (2003). The next step in Web services. *Communications of the ACM*, 46(10), 29-34.
- Huhns, M. (2002). Agents as Web services. *IEEE Internet Computing*, 6(4), 93-95.
- Jennings, N., Sycara, K., & Wooldridge, M. (1998). A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1(1), 7-38.
- Jose, R., Moreira, A., & Rodrigues, H. (2003). The AROUND architecture for dynamic location-based services. *Mobile Networks and Applications*, 8(4), 377-387.
- Karakasidis, A., & Pitoura, E. (2002). DBGlobe: A data-centric approach to global computing. *Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops (ICDCSW'2002)*, Vienna, Austria.
- Kuno, H., & Sahai, A. (2002). *My agent wants to talk to your service: Personalizing Web services through agents* (Technical Report No. HPL-2002-114). HP Laboratories, Palo Alto, California, USA.
- Maamar, Z., Ben-Younes, K., & Al-Khatib, G. (2003). Scenarios of supporting mobile users in wireless networks. *Proceedings of the 2nd International Workshop on Wireless Information Systems (WIS'2003) in*

*conjunction with the 5th International Conference on Enterprise Information Systems (ICEIS2003)*, Angers, France.

Papazoglou, M., & Georgakopoulos, D. (2003). Introduction to the special issue on service-oriented computing. *Communications of the ACM*, 46(10).

Ratsimor, O., Joshi, A., Finin, T., & Yesha, Y. (2003). eNcentive: A framework for intelligent marketing in mobile peer-to-peer environments. *Proceedings of the 5th International Conference on Electronic Commerce (ICEC'2003)*, Pittsburgh, Pennsylvania.

Valavanis, E., Ververidis, C., Vazirgiannis, M., Polyzos, G.C., & Norvag, K. (2003). MobiShare: Sharing context-dependent data & services from mobile sources. *Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence (WI'2003)*, Halifax, Canada.

## KEY TERMS

**Ad-Hoc Network:** It is a local area network or other small network, especially one with wireless or temporary plug-in connections, in which some of the network devices (sometimes mobile) are part of the network only for the duration of a communication session or because some of the devices are in some close proximity, so a communication session can take place.

**Context:** It is the information that characterizes the interaction between humans, applications, and the surrounding environment. Context can be decomposed into three categories: (i) computing context (e.g., network connectivity, communication cost); (ii) user context (e.g., user profile, location, nearby people); and (iii) physical context (e.g., lighting, noise levels).

**Mobile Support Station:** It is an equipment that manages mobile devices in terms of identifying their physical location and handling their incoming and outgoing messages/calls. A mobile support station always communicates with mobile users within its radio coverage area.

**Software Agent:** It is a piece of software that autonomously acts to carry out tasks on users' behalfs. In agent-based applications, it is accepted that users only need to specify high-level goals instead of issuing explicit instructions, leaving the decisions of how and when to their respective agent. A software agent exhibits a number of features that make it different from other traditional components including autonomy, goal-orientation, collaboration, flexibility, self-starting, temporal continuity,

character, communication, adaptation, and mobility. It should be noted that not all these features have to embody an agent.

**UDDI Registry:** The UDDI specifications define a way to publish and discover information on Web services. At a conceptual level, the information provided in a UDDI business registration consists of three components. First, the white pages component includes address, contact, and known identifiers. Second, the yellow pages component includes industrial categorization based on standard taxonomies. Finally, the green pages component includes the technical information about services that a business exposes. At a business level, the UDDI business registry can be used for checking whether a given partner has particular Web service interfaces, finding companies in a given industry with a given type of service, and locating information about how a partner or intended partner has exposed a Web service. The objective is to get aware of the technical details required for interacting with that service.

**Web Service:** It is an accessible application that other applications and humans can discover and trigger. The following properties define a Web service: (i1) independent as much as possible from specific platforms and computing paradigms; (ii2) developed mainly for inter-

organizational situations rather than for intra-organizational situations; and (iii3) easily composable so that its composition with other Web services does not require developing complex adapters.

**Wireless Environment:** More and more people are equipped with handheld devices such as cell phones, PDAs, and laptops. To manage all these devices, a computing model is deployed. It consists of two entities: mobile clients and fixed hosts. Some of the fixed hosts, called mobile support stations, are augmented with wireless interfaces. A mobile support station communicates with the mobile clients within its radio coverage area called wireless cell. Each cell has an identifier that is periodically broadcasted to each mobile client residing in the cell. The aim is to be fully aware of the mobile clients that are under the management of the cell. Mobile clients get information from information servers, through the mobile support stations. As long as a mobile client stays in the coverage area of the same mobile support station, there are no major issues in pushing information to the client's mobile device. As soon as the client leaves that coverage area, another mobile support system has to be in charge of the information transfer.

# Replication Methods and Their Properties

Lars Frank

Copenhagen Business School, Denmark

## INTRODUCTION

The most important evaluation criteria for replication methods are availability, performance, consistency, and costs. Performance and response time may be improved by substituting remote data accesses with local data accesses to replicated data. The availability of the system can be increased by using replicated data in case a local failure or disaster should occur. The major disadvantages of data replication are the additional costs of updating replicated data and the problems related to managing the consistency of the replicated data. Tables 1 and 2 give an overview of the evaluation of the replication methods described in this article. Frank (1999) described how such replication overviews may be used to optimize databases in practice. This article evaluates many more replication methods and therefore, it is possible to optimize even more. However, the evaluation criteria previously described have to be subdivided to illustrate the different properties of the different replication methods.

## BACKGROUND

Different versions of the 1-safe and 2-safe designs have been described in Garcia-Molina and Polyzois (1990); Polyzois and Garcia-Molina (1994); Gellersdörfer and Nicola (1995); and Humborstad, Sabaratnam, Hvasshovd, and Torbjornsen, (1997). The 0-safe design with local commit has been used in practice for a number of years and was described by Frank and Zahle (1998). The 2-safe design, the basic 1-safe design, the 1-safe design with commutative updates, and the 0-safe design with local commit have all been described and evaluated in detail by Frank (1999), and where it is possible, the evaluations will be reused in this article. The *countermeasure transaction model* is the first transaction model that systematically tries to describe how to eliminate or reduce the consistency and anomaly problems, which some replication methods may cause. This transaction model owes many of its properties to Garcia-Molina and Salem (1987), Mehrotra, Rastogi, Korth, and Silberschatz (1992), Weikum and Schek (1992), and Zhang, Nodine, Bhargava, and Bukhres (1994). Many new versions of the classic replication designs have

Table 1. Evaluation overview of some of the most important table replication designs

Evaluation criteria	DBMS supported replication methods					
	<i>n</i> -safe design with the ROWA protocol	<i>n</i> -safe design with the quorum protocol	1-safe design. Basic solution	1-safe design with commutative updates	0-safe design with local commit	0-safe design with deferred commit
Read performance/capacity	Best	Worst	Average	Average	Best	Best
Write performance	Worst	Above worst	Average	Average	Best	Below best
Ease of failure recovery	Average	Average	Worst	Average	Best	Best
Ease of disaster recovery	Best	Below best	Above worst	Average	Average	Average
Probability of lost data <sup>1</sup>	Best. $p^n$	Below best $p^{n+2}$	Worst $p$	Average	Average	Average
Logging of the update transaction <sup>2</sup>	Not supported	Not supported	Not supported	Recommended	Recommended	Recommended
Availability <sup>3</sup>	$1-q^n$	$1-q^n$	$1-q^n$	$1-q^n$	$1-q^n$	$1-q^n$
Atomicity	Best	Best	Worst	Best	Best	Best
Consistency	Best	Best	Average	Average	Worst	Worst
Isolation	Best	Best	Average	Average	Worst	Worst
Durability	Best	Best	Worst	Best	Best	Best
Development costs	Best	Best	Best	Average	Worst	Worst

been described in recent years. The most important versions are described after the corresponding classic design. How to use different replication methods to optimize and integrate ERP and e-commerce system have been described in detail by Frank (2004).

In the following, I will first describe the criteria used to evaluate the replication methods. Next, I will describe the most important replication methods used in practice. I do not prove that the evaluations illustrated in Tables 1 and 2 are correct, but arguments for the evaluations are presented. However, Frank (1999) has proved that the evaluations are correct for some of the described replication methods, and I believe that Frank's

Table 2. Evaluation overview of operating system replication methods

Evaluation criteria	Operating system replication methods			
	Mirroring with disk volume ownership	Mirroring without disk volume ownership	Remote caching in a fast storage	Local caching in the user location
Read performance/capacity	Average	Best	Average	Best
Write performance	Average	Above worst	Average	Average
Ease of failure recovery	Average	Average for roll back recovery	Average	Not supported
Ease of disaster recovery	Below best	Below best	Not supported	Not supported
The probability of lost data	Best $p^n$	Best $p^n$	Worst	Worst
Availability	$1-q^n$	$1-q^n$	$1-q^2$	$1-q^2$
Atomicity	Best	Not supported	Best	Not supported
Consistency	Best	Not supported	Best	Not supported
Isolation	Best	Not supported	Best	Not supported
Durability	Best	Not supported	Best	Not supported
Development costs	Best	Best	Best	Best

proof can be expanded to encompass all the replication methods described in this article.

## Description of the Evaluation Criteria for Replication Methods

In this article, the properties of the replication designs are normally described as *best*, *average*, *below average*, *worst*, and so forth, which allows us to compare the designs relatively. In general, it is not possible to select one of the replication designs as the best because all the designs have very different properties. However, if the requirements of an application are known, it is possible to select the most inexpensive design that fulfills the needs of the application. Next is a short description of the evaluation criteria used in this article.

The *read performance/capacity* property of a table design is evaluated to be *best* if remote readings always can be substituted by local readings. The *n*-safe design with the quorum protocol has the *worst* read performance/capacity because it is always necessary to lock a majority of the record copies to implement the isolation property.

The *write performance/capacity* property of a table design is evaluated to be *best* if a global table update always can be committed locally without communication with other locations. The *n*-safe design with the read one, write all (ROWA) protocol is evaluated to be *worst* in terms of write performance because it is always necessary to lock all the record copies to implement the isolation property.

After a local failure, the local database and its log files are not physically destroyed, and in contrast to database disasters, it is always possible to repair the site in such a way that operation may continue by using the failed site. *Ease of failure recovery* is evaluated to be *best* if the system automatically can make recovery without aborting all noncommitted transactions in case of a site failure. *Ease of failure recovery* is evaluated to be *average* if the system always has to abort noncommitted transactions in case of a site failure. The basic 1-safe design is evaluated to have the *worst* ease of failure recovery because even committed transactions may be lost after a failure in which a secondary copy is used as the primary copy (*lost transactions*).

A *database disaster* is as a situation in which a local database and its log files are destroyed. *Ease of disaster recovery* is evaluated to be *best* if it is possible to repair the database automatically. The no replication design is evaluated to be *worst* in terms of ease of disaster recovery because, at best, only an old, remote database copy can be used for recovery.

In case of a disaster, data may be lost. The probability

of avoiding lost data is evaluated to be better the lower the probability.

In case of a disaster, committed transactions may be lost, except in the *n*-safe designs. Therefore, the evaluation criteria, that is, ease of disaster recovery and availability, depend on the evaluation criterion, *logging of the update transactions in the locations of the clients*.

The *availability* of a database can be defined as the probability of having access to the database.

An update transaction is *atomic* if all or none of its updates are executed. The *atomicity* property of a table design is evaluated to be *best* if the database management systems (DBMSs) and the transaction model can guarantee the property, even in case of failure. The basic 1-safe design is evaluated to be *worst* in terms of the atomicity property because lost transactions may occur after the commitment of a transaction.

A database is *consistent* if the data in the database complies with some user-defined consistency rules. Transactions have by definition the *consistency property* if they fulfill the following condition:

“If a database is consistent when a transaction starts, the database must also be consistent when the transaction has been committed.”

The consistency property of a table design is evaluated to be *best* if the DBMSs and the transaction model can guarantee the property even in case of failure. The 0-safe designs are evaluated as *worst* in terms of consistency property, because in these designs the distributed database is normally inconsistent, and therefore only *asymptotic consistency* can be implemented (i.e., the database converges towards a consistent state).

Transactions are executed in *isolation* if the updates of each transaction can be seen only by other transactions after the updates have been committed or aborted. The isolation property of a table design is evaluated to be *best* if the DBMSs and the transaction model can guarantee the property even in case of failure. The 0-safe designs are evaluated to be *worst* in terms of isolation property because all types of isolation anomalies may occur if they are not managed by adopting countermeasures (Frank & Zahle, 1998).

The updates of transactions are said to be *durable* if they are stored in stable storage and secured by a log recovery system. The durability property of a table design is evaluated to be *best* if the DBMSs and the transaction model can guarantee the property even in case of failure. The basic 1-safe design is evaluated to be *worst* in terms of the durability property, because lost transactions may occur after the commitment of a transaction (i.e., the updates are not durable in case of a failure where a secondary copy is used as the primary copy).



The *development costs* of a table design are evaluated to be *best* if no special application programming is needed (i.e., all replication problems are managed by using DBMS tools). Therefore, the development costs of the  $n$ -safe designs, the no-replication design, and the basic 1-safe design have the *best* rating in Table 1. The development costs of the 0-safe designs are evaluated as *worst* because in these designs one has to select and implement countermeasures against the lost update anomaly, the dirty read anomaly, the nonrepeatable read anomaly, and the phantom anomaly (Berenson et al., 1995; Breibart, Garcia-Molina, & Silberschatz, 1992). The development costs of the 1-safe design with commutative updates are evaluated as *average* because in this design it is only necessary to implement countermeasures against the lost update anomaly.

### DESCRIPTION OF THE MOST IMPORTANT REPLICATION METHODS

In Table 1, the replication methods described by Frank (1999) are evaluated together with the main types of the  $n$ -safe design and a group of 0-safe designs that have deferred commit. This article will also evaluate replication designs that can be implemented as part of the operating system, and, therefore, cannot automatically support ACID properties.

All the  $n$ -safe designs described in this article can be optimized by using the so called atomic broadcast protocol (Hadzilacos & Toueg, 1993; Weismann, Pedone, Schiper, Kemme, & Alonso, 2000). This is a group communication technique in which messages from the coordinator are delivered in the same order in all the participating locations. This property can simplify the distributed concurrency control if the coordinator uses the atomic broadcast protocol to transfer the execution order in the coordinator location to the locations of the participants. This is possible if data are replicated in such a way that the coordinator can execute transactions locally and, only at the end, uses the atomic broadcast protocol to coordinate the update of replicated data. At the same time, performance is improved, as there is communication between the involved locations once per transaction and not once per data access. However, the improved performance possibilities are not reflected in Table 1, as the atomic broadcast protocol has not been implemented in any commercial DBMS product. All the  $n$ -safe designs described have *eager replication* (i.e., the updates of transactions are propagated to any replicated copies before the commit of the transactions).

Performance of the 1-safe and 0-safe designs can be optimized by *lazy replication*, by which the updates to any replicated copies are propagated after the commitment of the updating transaction. This improves the

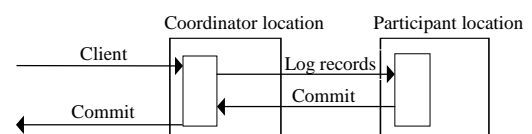
response time at the cost of the consistency between replicated data. The atomic broadcast protocol described cannot improve the performance of the lazy replication designs, as the transactions have already been committed when the update of replicated data takes place. However, the performance of the 1-safe designs and 0-safe designs can be improved in the same way as the  $n$ -safe designs by replicating data in such a way that transactions can run locally until commitment.

### The $n$ -Safe Design with the ROWA Protocol and the 2-Safe Design

The 2-safe design is a special case of the  $n$ -safe design, in which only two copies of a file exist. In the 2-safe design (Gray & Reuter, 1993), the primary/coordinating transaction manager involves the backup/participant system in the commit. If the participant system is up, the coordinating transaction manager sends the log records of the transaction at the end of commit phase 1. The coordinating transaction manager will not commit until the participant responds (or is declared down). The 2-safe design has good ratings in read capacity, consistency, isolation and easiness of failure/disaster recovery. The main problem is the poor write performance and a high risk of transaction restarts in case of major failures in one of the database locations or in the network connecting the locations. The 2-safe design may easily be generalized to the  $n$ -safe design, where  $n > 2$ . In the  $n$ -safe design, the coordinating transaction manager commits an update as a function of the  $n-1$  participating transaction managers' responses. There are many versions of the  $n$ -safe design. However, in all the versions it is possible to optimize the write performance if the participating transaction managers answer immediately without first forcing the log records to durable storage. This is acceptable in the  $n$ -safe design because, in case of a disaster, it is extremely unlikely that all  $n$  copies of the log are involved in the disaster. Anyway, the  $n$ -safe design still has a relatively low write performance.

ROWA means that in terms of reading, only one copy of the record is read, and in terms of writing, all the copies must be written/updated. The  $n$ -safe design with the ROWA protocol has the *best* read perfor-

Figure 1. 2-safe database design





mance, as a local record copy often can be used. The write performance is evaluated as *worst*, as all record copies must be locked before an update can be executed. The ROWA protocol is very vulnerable to both communication and site failures. The read one, write all available (ROWAA) protocol is a new version of the ROWA protocol. It is tolerant to both communication and site failures at the cost of controlling all participating copies available when the transaction is committed (e.g., Coulouris, Dollimore, & Kindberg, 2001). However, this extra control reduces the performance for all transactions. In stable networks, where failures are rare, it is therefore much better to restart the traditional ROWA protocol with a new value of  $n$  in case of communication or site failures.

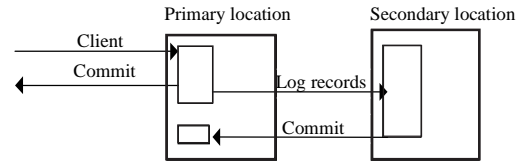
### The n-Safe Design with the Quorum Consensus Protocol

The *quorum consensus protocol* is tolerant to both communication and site failures, as only a number of locations with a quorum is necessary for accessing data. In the quorum consensus protocol, a number of votes is assigned to each copy. Each read operation must obtain a read quorum  $R$  before it can execute the read operation, and each write/update operation must obtain a write quorum  $W$  before it can execute the write/update. To obtain the isolation property,  $R$  and  $W$  must be selected in such a way that  $W$  is greater than half the amount of votes, and  $R + W$  are greater than the total number of votes. If  $R = 1$ ,  $W =$  the number of locations with a copy, and each location has one vote, the quorum consensus protocol specializes itself to the ROWA protocol, and therefore it is assumed in the following that  $R$  is greater than 2. In this case, the ROWA protocol has a *better* read performance than the quorum consensus protocol as the ROWA protocol only has to read one copy. The quorum consensus protocol has a better write performance than the ROWA protocol as the quorum consensus protocol only has to lock  $W$  copies before it is possible to write/update replicated data. The quorum consensus protocol provides better availability than the ROWA protocol as the quorum consensus protocol can tolerate some communication and site failures.

### The Basic 1-Safe Design

In the basic 1-safe design (Gray & Reuter, 1993), the primary transaction manager goes through the standard commit logic and declares completion when the commit record has been written to the local log. In a 1-safe design, the log records are asynchronously spooled to the locations of the secondary copies. In case of a primary site failure in the 1-safe design, production may

Figure 2. 1-safe database design

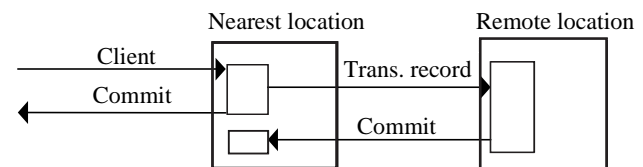


continue by selecting one of the secondary copies as a new primary copy. When the failure in the old primary copy location has been repaired, the log records from this location cannot always be used to update the new primary location (former secondary copy) because the records in the new primary copy may have been updated. *Lost transactions* are defined as the updates committed in the failed old primary copy and not in the new primary copy at the time production restarts in the new primary location. The main problem with the 1-safe design is that the lost transactions must be reconstructed and reexecuted before the recovery process is finished. Discontinuing production after a failure may prevent this problem. Therefore, selection of a new primary copy is only used in case of a disaster or a very serious failure.

### The 0-Safe Design with Local Commit

The *0-safe design with local commit* is defined as  $n$  table copies in different locations, where each transaction first will go to the nearest database location where it is executed and committed locally. If the transaction is an update transaction, the transaction propagates asynchronously to the other database locations where the transaction is reexecuted without user dialogue and committed locally at each location. This means that all the table copies normally are inconsistent and not up to date under normal operation. The inconsistency must be managed by using countermeasures against the isolation anomalies. For example, to prevent lost updates in the 0-safe design, all update transactions must be designed to be commutative (Frank & Zahle, 1998). From a performance and disconnection point of view, the 0-safe de-

Figure 3. 0-safe database design



sign with local commit is the best choice and, therefore it is recommended for mobile computing. Other versions of the 0-safe design where commitment is deferred are described later.

### The 1-Safe Design with Commutative Updates

The 1-safe design can transfer updates to the secondary copies in the same way as the 0-safe design. In this design, lost transactions cannot occur because the transaction transfer of the 0-safe design makes the updates commutative. The properties of this mixed design will come from either the 1-safe or the 0-safe design. The 1-safe design with commutative updates does not have the high update performance and capacity of the 0-safe design. On the other hand, in this design the isolation property may be implemented automatically as long as the primary copy does not fail. Normally, this makes it much cheaper to implement countermeasures against the isolation anomalies, because it is only necessary to secure that the lost transactions are not lost in case of a primary copy failure.

### The 0-Safe Designs with Deferred Commit

The *0-safe designs with deferred commit* are defined as a group of replication methods in which a global transaction first must update the local copy closest to the user by using a compensatable subtransaction. Later, the local update may be committed globally by using one of the following designs:

- *0-safe design with primary copy commit*, in which the global transaction may be committed/rejected in a (remote) primary copy location. If the update of the primary copy is committed, retrievable subtransactions will be propagated to the rest of the secondary copy locations, which must be updated and committed sooner or later. However, if the update of the primary copy is rejected, a compensating subtransaction must be propagated to the location where the transaction was first created.
- *0-safe design with all participants commit*, in which the global transaction may be committed if all the participating locations with local copies can accept the original compensatable updating.
- *0-safe design with token commit*, in which the global transaction may be committed when the user who initiated the compensatable

subtransaction receives a “token” that excludes all other users from committing global transactions.

R

### Mirroring

*Mirroring* techniques enable the operating system (eventually supported by hardware) to write the data simultaneously on different disks in such a way that the data is replicated two or more times, as needed. Mirroring methods may be grouped the same way as the *n*-safe, 1-safe and 0-safe designs. However, this article only evaluates *n*-safe mirroring methods, as these methods have the most interesting properties compared to the corresponding *n*-safe designs. Mirroring replication methods can also be grouped according to the processes that can access data under normal operations. In mirroring with disk-volume ownership, a single concurrency managing process owns a disk volume and its mirrored data as long as the process is operative. Therefore, read performance is *average*, as all reads must go through this process. However, in this situation it is also possible to optimize write performance to *average*, as distributed locking is not necessary. Failure recovery is the same as in a central database, except that a database copy may fail without any problems. Therefore, it does not matter that mirroring does not support ACID properties in this situation. In mirroring without disk-volume ownership, concurrent processes with their own local DBMS may lock data in a ROWA-like fashion, and therefore the read performance is *best* and the write performance close to *worst*.

### Caching

*Cached data* is a snapshot of some frequently used data. Normally, cached data is incomplete, and therefore it is not suited for disaster recovery. In *remote caching*, a primary copy of the frequently used data is normally stored on a very fast medium to optimize access to data. In *local caching*, an often-inconsistent secondary copy of the frequently used data is stored in or close to the location of some users to optimize their access to the data. The local cache may be updated periodically or when a local user makes an update. Therefore, the local cache is often inconsistent.

### FUTURE TRENDS

Many new versions of the classic replication designs have been described in recent years, and I believe that this process will continue. However, a new important trend is that hybrid replication protocols (e.g., see Irún, Muñoz, & Bernabéu, 2003) can be developed in such a

way that they behave as different replication methods, depending on the system configuration or the application requirements. However, such solutions are not yet available in commercial database management systems.

## CONCLUSION

To my knowledge, this article is the first to evaluate all the most commonly used replication methods. The replication methods are grouped into types, and two methods within each group are evaluated. Tables may be fragmented, and replication can be optimized by selecting the best replication method for each fragment. However, inconsistencies and anomalies may complicate the selection of replication methods, as countermeasures against anomalies should be selected per application/system and not per fragment, as described by Frank and Zahle (1998).

## REFERENCES

Berenson, H., Bernstein, P., Gray, J., Melton, J., O'Neil, E., & O'Neil, P. (1995). A critique of ANSI SQL isolation levels. *Proceedings of the ACM SIGMOD Conference*.

Breibart, Y., Garcia-Molina, H., & Silberschatz, A. (1992). Overview of multidatabase transaction management. *VLDB Journal*, 2, 181-239.

Coulouris, G., Dollimore, J., & Kindberg, T. (2001). *Distributed systems concepts and design*. Addison-Wesley.

Frank, L. (1999). Evaluation of the basic remote backup and replication methods for high availability databases. *Software - Practice & Experience*, 29(15), 1339-1353.

Frank, L. (2004). Architecture for integration of distributed erp systems and e-commerce systems. *Industrial Management and Data Systems*, 104(5), 418-429.

Frank, L., & Zahle, T. (1998). Semantic ACID properties in multidatabases using remote procedure calls and update propagations. *Software - Practice & Experience*, 28, 77-98.

Gallersdörfer, R., & Nicola, M. (1995). Improving performance in replicated databases through relaxed coherency. *Proceedings of the 21st VLDB Conference*.

Garcia-Molina, H., & Polyzois, C. (1990). Issues in disaster recovery. *IEEE Compton*. (pp. 573-577). New York: IEEE.

Garcia-Molina, H., & Salem, K. (1987). Sagas. *Proceedings of the ACM SIGMOD Conference*.

Gray, J., & Reuter, A. (1993). *Transaction processing*. Morgan Kaufman.

Hadzilacos, V., & Toueg, S. (1993). Fault-tolerant broadcasts and related problems. In S. Mullender (Ed.), *Distributed systems* (pp. 97-145). Addison-Wesley.

Humborstad, R., Sabaratnam, M., Hvasshovd, S., & Torbjornsen, O. (1997). 1-Safe algorithms for symmetric site configurations. *Proceedings of the 23rd VLDB Conference*.

Irún, L., Muñoz, F. D., & Bernabéu, J. (2003). An improved optimistic and fault-tolerant replication protocol. *Lecture Notes in Computer Science*, 2822, 188-200.

Mehrotra, S., Rastogi, R., Korth, H., & Silberschatz, A. (1992). A transaction model for multi-database systems. *Proceedings of the International Conference on Distributed Computing Systems*.

Polyzois, C., & Garcia-Molina, H. (1994). Evaluation of remote backup algorithms for transaction-processing systems. *ACM TODS*, 19(3), 423-449.

Weikum, G., & Schek, H. (1992). Concepts and applications of multilevel transactions and open nested transactions. In A. Elmagarmid (Ed.), *Database transaction models for advanced applications* (pp. 515-553). Morgan Kaufmann.

Wiesmann, M., Pedone, F., Schiper, A., Kemme, B. & Alonso, G. (2000). Understanding replication in databases and distributed systems. *Proceedings of the 20th International Conference on Distributed Computing Systems*, Taipei, Taiwan, Republic of China.

Zhang, A., Nodine, M., Bhargava, B., & Bukhres, O. (1994). Ensuring relaxed atomicity for flexible transactions in multidatabase systems. *Proceedings of the ACM SIGMOD Conference*.

## KEY TERMS

**0-Safe Designs:** Replication methods where none of the copies are consistent and up-to-date.

**1-Safe Designs:** Replication methods where 1 primary copy is consistent and up-to-date.

**Availability of Data:** The probability of having access to the data. Replication will normally increase data availability.

**Caching:** Replication methods where access to frequently used data is optimized. In *remote caching*, a

## Replication Methods and Their Properties

primary copy of the frequently used data is normally stored on a very fast medium to optimize access to data. In *local caching*, an often inconsistent secondary copy of the frequently used data is stored in or close to the location of some users to optimize their access to the data.

**Disaster Recovery:** Recovery in a situation where both the current database and its log-files have been destroyed.

**Failure Recovery:** Recovery in a situation where log-files and the current database or an old database copy are available.

**Mirroring:** Replication methods where all updates to a logical disk volume are copied by the operating system to two or more physical disk volumes.

**n-Safe Designs:** Replication methods where all  $n$  copies are consistent and up-to-date.

**Read Performance:** The maximum number of read operations per time unit. Replication will normally increase read performance.

**Replication Method:** A method for managing redundant data in such a way that a system can be optimized in some way.

**Write Performance:** The maximum number of write operations per time unit. Replication will normally decrease write performance and give rise to a possible of inconsistency between the replicated data.

R

## ENDNOTES

- <sup>1</sup> The probability of lost data as a function of the probability, say  $p$ , of a local disaster.
- <sup>2</sup> Logging of the update transactions in the locations of the clients.
- <sup>3</sup> The availability as a function of the probability, say  $q$ , of a local site failure.

# Rewriting and Efficient Computation of Bound Disjunctive Datalog Queries

**Sergio Greco**

*DEIS Università della Calabria, Italy*

**Ester Zumpano**

*DEIS Università della Calabria, Italy*

## INTRODUCTION

A strong interest in enhancing Datalog programs by the capability of disjunction emerged in several areas, such as databases, artificial intelligence, logic programming, and so forth (Lobo, Minker & Rajasekar, 1992). Disjunctive rules have been profitably used in several contexts including knowledge representation, databases querying, and representation of incomplete information (Eiter, Gottlob & Mannila, 1997a; Gelfond & Lifschitz, 1991). Most of the works on disjunction are concerned with the definition of intuitive and expressive semantics, which are commonly based on the paradigm of minimal models (Abiteboul, Hull & Vianu, 1995; Lobo et al., 1992; Ullman, 1989).

Disjunction in the head of rules increases the expressive power of the language but makes the computation of queries very difficult, as it allows the presence of multiple models whose number is generally exponential with respect to the size of the input (Abiteboul et al., 1995; Eiter et al., 1997a).

Therefore, a great interest has been devoted to the definition of efficient algorithms for both computing the semantics of programs and answering queries.

Most of the research has concentrated on the definition of efficient fixpoint algorithms computing the semantics of programs, and the more effective proposals are based on the evaluation of the (intelligent) ground instantiation of programs and on the use of heuristics (Leone, Rullo & Scarcello, 1997). However, different from standard datalog, for disjunctive queries, there are few effective methodologies which systematically utilize the query to propagate bindings into the body of the rules to avoid computing all the models of the program.

The following example taken from Greco (2003), shows a program in which only a strict subset of the minimal models needs to be considered to answer the query.

**Example 1.** Consider the disjunctive program P1 consisting of the following rule:

$$p(X) \vee q(X) \leftarrow a(X, Y)$$

and a database D consisting of the set of facts  $\{a(1,2), a(2,3), \dots, a(k, k+1)\}$ . Consider now a query asking if there is some model for  $P \cup D$  containing the atom  $p(1)$ . A “brute force” approach, based on an exhaustive search of the minimal models of  $P \cup D$ , would consider  $2^k$  minimal models.

However, to answer the query, we could only consider the ground rule:  $p(1) \vee q(1) \leftarrow a(1,2)$  and, therefore, only two minimal (partial) (for partial model we intend a subset of the model containing all the atoms necessary to answer the query) models:  $M1 = \{p(1)\} \cup D$  and  $M2 = \{q(1)\}$ .

From the above example, it is evident that using the query goal to reduce the size and the number of models to be considered for answering the query is necessary.

Constraints play an active role in the deductive database process as they model the interaction among data and define properties that have to be satisfied by models.

Thus, this article explores the efficient computation of bound queries over disjunctive Datalog programs enriched with constraints.

The technique here proposed is very relevant for the optimization of queries expressing hard problems. Indeed, the usual way of expressing declaratively hard problems, such as NP problems and problems in the second level of the polynomial hierarchy, is based on the *guess-and-check* technique, where the *guess* part is expressed by means of disjunctive rules, and the *check* part is expressed by means of constraints. However, as shown by the following example, the presence of constraints reduces the number of feasible models but could increase both the number of ground rules and the number of (partial) models to be considered for answering the query.

**Example 2.** Consider the query goal  $p(1)$  over the disjunctive program P2 consisting of the rule in the above example plus the following constraint (a rule with empty head which is true if the body is false)

$$\leftarrow p(X), a(X, Y), q(Y), X \leq 1.$$



To answer the query, we have to consider other than the ground rule:

- r1:  $p(1) \vee q(1) \leftarrow a(1,2)$ .  
 also the ground rule:  
 r2:  $p(2) \vee q(2) \leftarrow a(2,3)$ .

as the atom  $p(1)$  in the head of the rule r1 is influenced by the atom  $q(2)$  through the ground constraint c1

- $p(1), a(1,2), q(2), 1 \leq 1$ .

Consequently, there are three stable models to be considered:

- $N1 = \{ p(1), p(2) \} \cup D$ ,  
 $N2 = \{ q(1), p(2) \} \cup D$ , and  
 $N3 = \{ q(1), q(2) \} \cup D$ .

The interpretation  $\{ p(1), q(2) \} \cup D$  is not a model as it does not satisfy the constraint.

The above example shows the necessity of an effective methodology that uses the properties specified by the constraints and the query goal in order to compute correct answers avoiding the evaluation of useless models. However, in the presence of constraints, we have to consider ground rules defining atoms on which the query goal depends on (both through standard rules and constraints), but also to check that constraints are satisfied by the whole set of ground rules.

## BACKGROUND

In the literature, different approaches have been proposed for the efficient bottom-up evaluation of queries, for example, the Magic set (Bancilhon et al., 1986), the Counting (Beeri & Ramakrishnan, 1991), and the supplementary magic set and other specialized rewriting techniques (Beeri & Ramakrishnan, 1991; Greco, Saccà & Zaniolo, 1995; Ramakrishnan et al., 1993; Ullman, 1989). The key idea of all these techniques consists in the rewriting of deductive rules with respect to the query goal to answer the query without actually computing irrelevant facts. In this section, we recall the magic-set rewriting technique for Datalog queries as its generality and efficiency makes of this approach a kind of standard in the field.

Although the magic-set can be applied to general Datalog queries, for the sake of simplicity, we present here the technique for linear programs, that is, programs whose rules contain at most one body predicate mutually recursive with the head predicate.

The magic-set method consists of three separate steps:

- An *Adornment step* in which the relationship between a bound argument in the rule head and the bindings in the rule body is made explicit.
- A *Generation step* in which the adorned program is used to generate the *magic rules* which simulate the top-down evaluation scheme.
- A *Modification step* in which the adorned rules are modified by the magic rules generated in Step 2; these rules will be called *modified rules*.

An *adorned program*  $P^\beta$  is a program whose predicate symbols have associated a string  $\alpha$  defined on the alphabet  $\{b, f\}$  of length equal to the arity of the predicate. A character  $b$  (resp.  $f$ ) in the  $i$ -th position of the adornment associated with a predicate  $p$  means that the  $i$ -th argument of  $p$  is bound (resp. free).

The adornment step consists in generating a new program whose predicates are adorned. Given a rule  $r$  and an adornment  $\alpha$  of the rule head, the adorned version of  $r$  is derived as follows:

- Identify the distinctive arguments of the rules: an argument is distinctive if it is bound in the adornment  $\alpha$ , is a constant or appears in a base predicate of the rule-body which includes an adornment argument;
- Assume that the distinctive arguments are bound and use this information in the adornment of the derived predicates in the rule body.

Adornments containing only  $f$  symbols can be omitted.

Given a query  $Q = \langle q(T), P \rangle$  and letting  $\alpha$  be the adornment associated with  $q(T)$ , the set of adorned rules for  $Q$  is generated by (1) computing the adorned version of the rules defining  $q$  and (2) generating, for each new adorned predicate  $p^\alpha$  introduced in the previous step, the adorned version of the rules defining  $p$  w.r.t.  $\alpha$ ; Step 2 is repeated until no new adorned predicate is generated.

The second step of the process consists in using the adorned program for the generation of the magic rules. For each of the adorned predicates in the body of the adorned rule:

- Eliminate all the derived predicates in the rule body which are not mutually recursive with the rule head;
- Replace the derived predicates symbol  $p^\alpha$  with  $\text{magic\_}p^\alpha$  and eliminate the variables which are free w.r.t.  $\alpha$ ;
- Replace the head predicates symbol  $q^\alpha$  with  $\text{magic\_}q^\alpha$  and eliminate the variables which are free w.r.t.  $\alpha$ ; and

- Interchange the transformed head and derived predicate in the body.

Finally, the modification step of an adorned rule is performed as follows: for each adorned rule whose head is  $p^\alpha(X)$ , where  $X$  is a list of variables, extend the rule body with  $\text{magic\_p}^\alpha(X')$  where  $X'$  is the list of variables in  $X$  which are bound w.r.t.  $\alpha$ .

**Example 3.** Consider the query  $Q_3 = \langle \{p(1,C)\}, P_3 \rangle$  where  $P_3$  is defined as:

$$\begin{aligned} p(X,C) &\leftarrow q(X,2,C). \\ q(X,Y,C) &\leftarrow a(X,Y,C). \\ q(X,Y,C) &\leftarrow b(X,Y,Z,W), q(Z,W,D), c(D,C). \end{aligned}$$

The adorned version of  $P_3$  w.r.t. the query goal  $p(1,C)$  is:

$$\begin{aligned} p^{\text{bf}}(X,C) &\leftarrow q^{\text{bbf}}(X,2,C). \\ q^{\text{bbf}}(X,Y,C) &\leftarrow a(X,Y,C). \\ q^{\text{bbf}}(X,Y,C) &\leftarrow b(X,Y,Z,W), q^{\text{bbf}}(Z,W,D), c(D,C). \end{aligned}$$

The rewritten query is  $Q'_3 = \langle p^{\text{bf}}(1,C)\rangle, P'_3 \rangle$  where  $P'_3$  is as follows:

$$\begin{aligned} &\text{magic\_p}^{\text{bf}}(1). \\ &\text{magic\_q}^{\text{bbf}}(X,2) \leftarrow \text{magic\_p}^{\text{bf}}(X). \\ &\text{magic\_q}^{\text{bbf}}(Z,W) \leftarrow \text{magic\_q}^{\text{bf}}(X,Y), b(X,Y,Z,W). \\ \\ &p^{\text{bf}}(X,C) \leftarrow \text{magic\_p}^{\text{bf}}(X), q^{\text{bbf}}(X,2,C). \\ &q^{\text{bbf}}(X,Y,C) \leftarrow \text{magic\_q}^{\text{bbf}}(X,Y), a(X,Y,C). \\ &q^{\text{bbf}}(X,Y,C) \leftarrow \text{magic\_q}^{\text{bbf}}(X,Y), b(X,Y,Z,W), q^{\text{bbf}}(Z,W,D), c(D,C). \end{aligned}$$

Note that the first set of rules consists of the magic rules generated in the second step, while the second set of rules consists of the modified rules.

Observe that, although the technique here presented applies only to negation free linear programs, the magic-set rewriting is general and can also be applied to nonlinear programs with some form of negation (e.g., stratified negation) where bindings are also propagated through derived predicates (Beeri & Ramakrishnan, 1991).

Let  $Q = \langle G, P \rangle$  be a query, then  $\text{Magic}(Q)$  denotes the query derived from  $Q$  by applying the magic-set method. The query  $\text{Magic}(Q)$  will be also denoted as  $\langle G^\alpha, \text{magic}(G,P) \rangle$  where  $G^\alpha$  denotes the adorned version of  $G$ , whereas  $\text{magic}(G,P)$  denotes the rewriting of  $P$  w.r.t. the goal  $G$ .

The rewritten program consists of two distinct sets of rules: a set of new rules (generated in Step 2), called *magic rules*, and the set of *modified rules* (generated in Step 3) which is derived from the set of rules in the source

program. The adorned rules generated in Step 1 are denoted by  $\text{Adorn}(G,P)$ .

## BINDING PROPAGATION AND CONSTRAINTS

In this section, we present a technique for propagating bindings into Datalog<sup>U</sup> queries with *strong* constraints, that is, classical constraints that have to be satisfied in each model. A constraint is a simple and powerful form of unstratified negation and every constraint of the form  $\leftarrow B$  is equivalent, under total stable model semantics to a standard rule of the form  $p \leftarrow \neg p, B$  where  $p$  is a new predicate symbol. The advantage of using constraints instead of unstratified negation is that they are highly expressive (disjunctive Datalog with constraints has the same expressive power of disjunctive Datalog with unstratified negation (Flesca et al., 2003)), and they have a simple and intuitive semantics (programs with unstratified negation are neither intuitive nor easily optimizable).

Despite standard Datalog, in which we propagate bindings from the head of rules into the body, the problem with constraint rules is that we need the bindings to be propagated also through the constraints. For instance, if we are interested in knowing whether  $p(1)$  is true in a program where there is the constraint  $\leftarrow p(X), q(X)$ , then we also need to evaluate the truth value of  $q(1)$ .

Thus, the truth value of each ground atom in a constraint depends on the truth value of the other atoms appearing in the same ground constraint, and, hence, in a more abstract perspective, constraint rules behave in a similar manner as disjunctive rules when propagating bindings into their heads.

In the following, a Datalog program  $P$  will be denoted by a pair  $(P_R, P_C)$ , where  $P_R$  is a not the empty set of positive disjunctive rules, and  $P_C$  is a set of positive constraints; moreover, by following the same guidelines of the previous section, we shall introduce the technique and the main results by using a running example.

**Example 4.** We are given the query  $\langle 2\text{col}(1,2), \text{Coloring} \rangle$ , checking whether a graph is 3-colorable and whether colors *red* and *blue* can be assigned to nodes 1 and 2 respectively. The program **COLORING** consists of the following rules:

$$\begin{aligned} 2\text{col}(X,Y) &\leftarrow \text{color}(X,\text{red}), \text{color}(Y,\text{blue}). \\ \text{color}(X,\text{red}) \vee \text{color}(X,\text{blue}) \vee \text{color}(X,\text{yellow}) &\leftarrow \text{node}(X). \\ \leftarrow \text{edge}(X,Y), \text{color}(X,C), \text{color}(Y,C) \end{aligned}$$

Given a set of constraints  $P_c$ , we denote with  $esv(P_c)$  the set of Datalog rules obtained by replacing each constraint in  $P_c$  having the form

$$\leftarrow a_1, \dots, a_k, b_1, \dots, b_m, \neg c_1, \dots, \neg c_m$$

where  $a_1, \dots, a_k$  are base atoms and  $b_1, \dots, b_m$  are derived atoms, with the following set of rules:

$$b_i \leftarrow b_1, \dots, b_{i-1}, b_{i+1}, \dots, b_m, a_1, \dots, a_k \quad i \in [1..m]$$

Moreover, given a disjunctive program  $P = (P_R, P_C)$ , the *extended standard version* of  $P$ , denoted by  $esv(P) = esv(P_R, P_C) = esv(P_R) \cup esv(P_C)$ .

**Example 5.** The extended standard version of the program **COLORING** of Example 3, obtained by the rewriting of constraints and disjunctive rules and denoted by  $esv(\mathbf{COLORING})$ , is as follows:

$$\begin{aligned} r1: & 2col(X, Y) \leftarrow color(X, red), color(Y, blue). \\ s1: & color(X, red) \leftarrow node(X). \\ s2: & color(X, blue) \leftarrow node(X). \\ s3: & color(X, yellow) \leftarrow node(X). \\ s4: & color(X, red) \leftarrow color(X, blue), node(X). \\ s5: & color(X, blue) \leftarrow color(X, red), node(X). \\ s6: & color(X, red) \leftarrow color(X, yellow), node(X). \\ s7: & color(X, yellow) \leftarrow color(X, red), node(X). \\ s8: & color(X, blue) \leftarrow color(X, yellow), node(X). \\ s9: & color(X, yellow) \leftarrow color(X, blue), node(X). \\ c1: & color(X, C) \leftarrow edge(X, Y), color(Y, C). \\ c2: & color(Y, C) \leftarrow edge(X, Y), color(X, C). \end{aligned}$$

where the rule  $r1$  is derived from  $r$ , the rules  $s1$ - $s9$  are derived from the rule  $s$ , and the rules  $c1$  and  $c2$  are derived from the constraint  $c$ .

Moreover, we denote by  $ESV(P)$  the program derived from  $esv(P)$  by replacing each derived predicate symbol  $g$  with a new predicate symbol  $G$ .

**Definition 1.** Let  $P = (P_R, P_C)$  be a disjunctive Datalog program. The program  $Rew(P)$  is defined as  $RV(P_R) \cup P_C \cup ESV(P)$ . Given a query  $Q = \langle g(t), P \rangle$ , we denote with  $Rew(Q)$  the query  $\langle g(t), Rew(P) \rangle$ .

**Example 6.** The set of restricted rules in  $RV(\mathbf{COLORING})$ , derived from the program **COLORING** of Example 3, is as follows:

$$\begin{aligned} 2col(X, Y) & \bullet 2COL(X, Y), \quad color(X, red), \\ & color(Y, blue). \\ & color(X, red) \vee color(X, blue) \vee color(X, yellow) \bullet \\ & node(X), \quad COLOR(X, red), \quad COLOR(X, blue), \\ & COLOR(X, yellow). \end{aligned}$$

where the predicates  $2COL$  and  $COLOR$  are defined in  $ESV(\mathbf{COLORING})$  which is derived from the program  $esv(\mathbf{COLORING})$ , reported in Example 5, by replacing  $2col$  with  $2COL$  and  $color$  with  $COLOR$ .

The complete rewriting of the program **COLORING** consists of the above rules plus the constraint:

- $edge(X, Y), color(X, C), color(Y, C)$ .

and the rules in  $ESV(\mathbf{Coloring})$ , presented in Example 5.

The first interesting result is that (as for Datalog programs without constraints) the above rewriting method does not affect the semantics of the query.

**Proposition 1.** Let  $P = (P_R, P_C)$  be a disjunctive Datalog program and  $Rew(P)$  be the rewritten version of  $P$ . Then, for every atom  $g(t)$  is  $\langle g(t), P \rangle \equiv \langle g(t), Rew(P) \rangle$ .

Thus, a viable way for reducing the number of models to be computed, by limiting the attentions to the ones that are really needed for answering the query, is to consider a program different w.r.t.  $ESV(P)$ , which is able to make an effective and sound restriction.

**Definition 2.** Let  $Q = \langle g(t), P \rangle$  with  $P = (P_R, P_C)$  be a disjunctive Datalog query, then the disjunctive magic-set rewriting of  $P$  w.r.t.  $g(t)$ , denoted by  $Disj\_Magic(g(t), P)$ , is equal to  $RV(P_R) \cup P_C \cup Coll(ESV(P), Magic(G(t), ESV(P))) \cup Magic(G(t), ESV(P))$ .

## EXPERIMENTAL RESULTS

In this section, we report some experimental results to give an idea of the improvements which can be obtained by means of our technique; the proposed algorithm has been included into the prototype presented in Greco (2003), and all the experiments have been carried out by means of the DLV system (Eiter et al., 1997b) on a PC with a Pentium 4, 1.7 GHz, and 512 Mbyte of RAM under the operating system Linux.

For the following queries, we have used graphs having the structure depicted in Figures 1(a) and 1(b) with  $base = height$ . Here,  $base$  denotes the number of nodes in the same layer,  $height$  the number of nodes in the same column, and  $output$  (input)  $grade$  the number of arcs starting from (ending at) each node. The number of nodes in the graph is  $base * height$  whereas the number of arcs is  $(base-1) * (height-1) * grade + (base-1) + (height-1)$  for the undirected graph  $G1$  and  $base * (height-1) * grade$  for the directed graph  $G2$ . We also denote with  $G'1$  a graph whose structure is derived from  $G1$  by adding a dashed arc as reported in Figure 1(a).

Figure 1 (a).

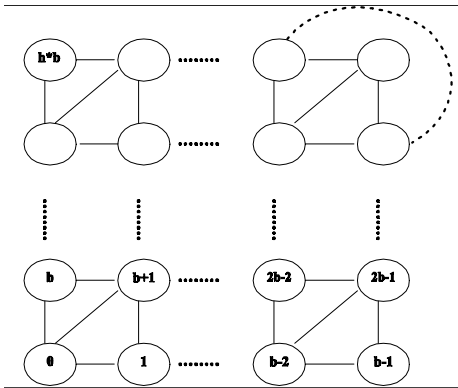


Figure 1 (b).

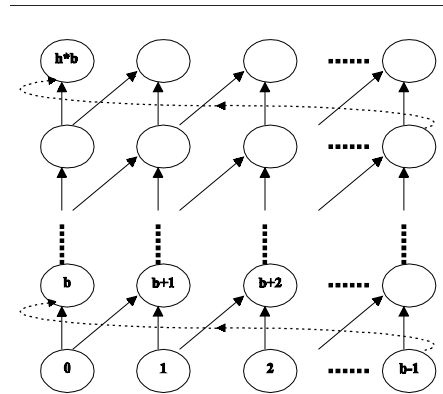


Figure 2 (a).

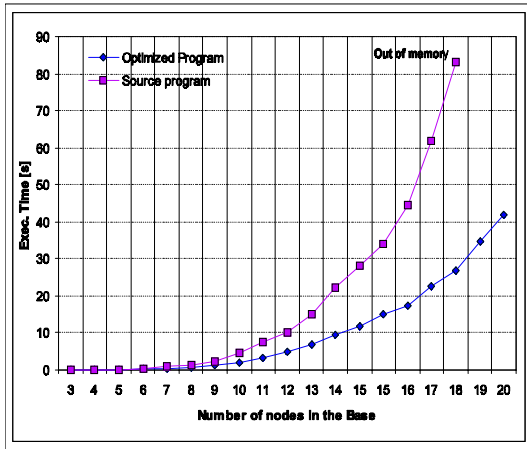


Figure 2 (b).

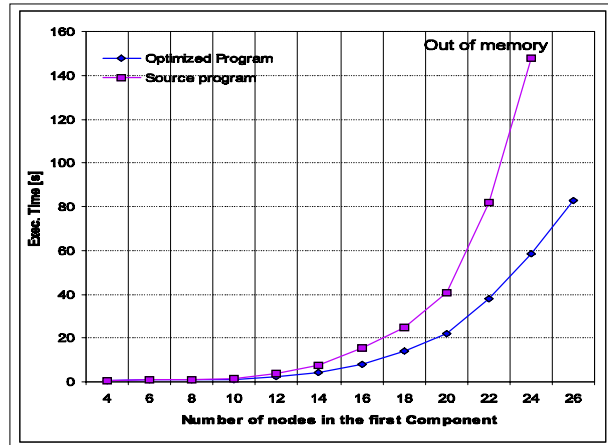


Figure 3 (a).

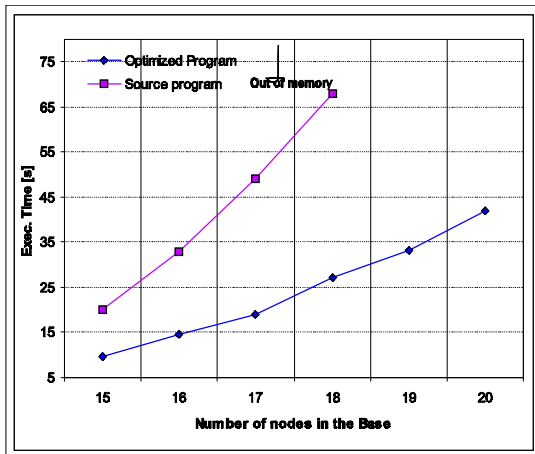


Figure 3 (b).

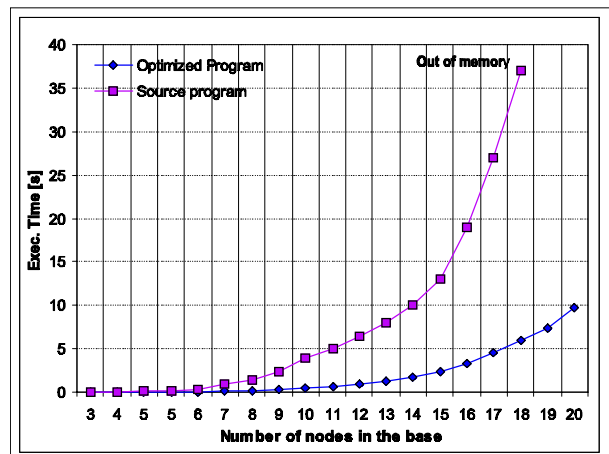


Figure 4(a).

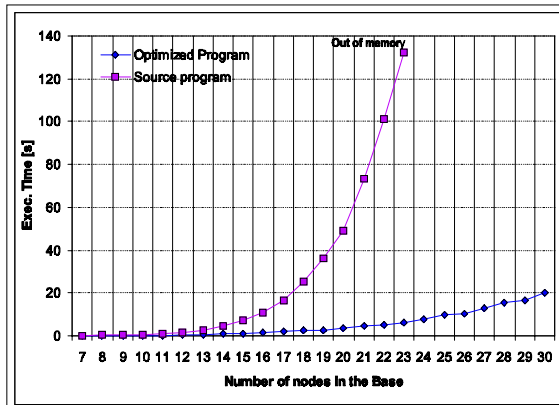
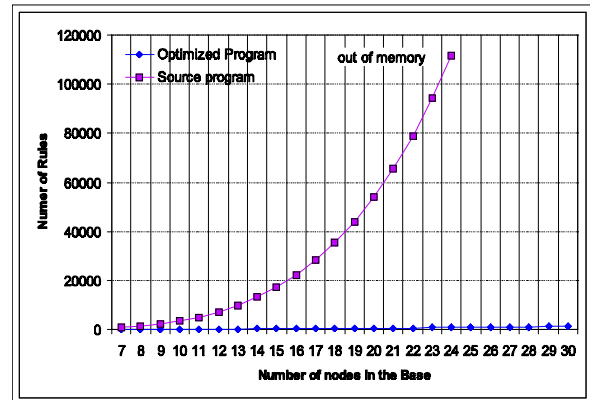


Figure 4(b).



**Example 7. 3-COLORING.** We consider the query of Example 3 with input graphs consisting of two disconnected components with variable sizes and having the structure shown in Figure 2(a). We have assumed that the nodes appearing in the query goal belong to the same components; that is, the binding is propagated only on one component. In the following, we denote as first component the one on which the binding is propagated and, as second component, the one on which the binding is not propagated. The results are shown in Figure 2, where we have considered colorable graphs and, in Figure 3, where we have considered not colorable graphs. In Figure 2(a), we have considered graphs with two components with the same size, ranging from  $3 * 3$  to  $20 * 20$  nodes. In Figure 2(b), we have considered graphs with two components having different sizes: the number of nodes in the first component ranges from  $4 * 4$  to  $26 * 26$  while the number of nodes in the second component ranges from  $4/2 * 4/2$  to  $26/2 * 26/2$ .

**Example 8. Simple Path.** The following example presents a query computing a simple path in a graph with start and end node fixed. A simple path is a path passing through the nodes once; a simple path is said to be hamiltonian if it passes through all nodes in the graph.

The first rule defines a partition of arcs into  $e$  and  $ne$ ; the rules defining  $sp$  compute paths on the graph defined by  $e$  (selected edges); and the last two rules state that every simple path cannot contain two distinct nodes  $Y$  and  $Y1$  (resp.  $X$  and  $X1$ ) connected to the same node  $X$  (resp.  $Y$ ).

We have used instances for the relation  $edge$  as depicted in Figure 2(b) with  $base = height$  and  $output\ grade = 2$ .

The results of the query are reported in Figure 4 where the x-axis shows the size of the graph. In more details,

Figure 3(a) reports the time to execute the query, whereas Figure 3(b) reports the number of disjunctive rules in the ground program. Also in this case, it results evident the great improvement provided by the proposed technique which ensures, for large graphs, more than one order of magnitude between the source and the optimized queries.

## CONCLUSION

The main contribution of the paper consisted in extending previous techniques for query optimization to Datalog programs with both disjunctive heads and constraints.

The proposed approach consisted of two distinct phases: the rewriting of queries for propagating bindings from the query goal into the program and the use of specialized algorithms computing rewritten queries. We have shown that the technique is sound and complete and can be profitably used for search and optimization problems. Several experiments have confirmed the validity of our technique.

## FUTURE TRENDS

A great interest has been devoted by the research community to the definition of efficient algorithms for both computing the semantics of programs and answering queries. This paper has shown the necessity for disjunctive queries of effective methodologies which systematically utilize the query to propagate bindings into the body of the rules to avoid the computation of all the models of the program. We are currently investigating an extension of the proposed technique to more flexible and powerful forms of constraints.



## REFERENCES

- Abiteboul, S., Hull, R., & Vianu, V. (1995). *Foundations of databases*. Addison-Wesley.
- Bancilhon, F., Mayer, D., Sagiv, Y., & Ullman, J.D. (1986). Magic sets and other strange ways to implement logic programs. *Proceedings of the PODS Conference*, 45(1-3) (pp. 1-16).
- Beeri, C., & Ramakrishnan, R. (1991). On the power of magic. *JLP*, 10(1/2/3/4), 255-299.
- Brass, S., & Dix, J. (1994). A general approach to bottom-up computation of disjunctive semantics. *Proceedings of the ICLP '94 W. on NELP* (pp. 127-155).
- Buccafurri, F., Leone, N., & Rullo, P. (2000). Enhancing disjunctive datalog by constraints. *IEEE TKDE*, 12(5), 845-860.
- Eiter, T., Gottlob, G., & Mannila, H. (1997a). Disjunctive datalog. *ACM TODS*, 22(3), 364-418.
- Eiter, T., Leone, N., Mateis, C., Pfeifer, G., & Scarcello, F. (1997b). A deductive system for non-monotonic reasoning. *LPNMR*, 363-374.
- Fernandez, J.A., & Minker, J. (1995). Bottom-up computation of perfect models for disjunctive theories. *JLP*, 25(1), 33-51.
- Flesca, S., Greco, S., Trubitsyna, I., & Zumpano, E. (2003). *On the semantics and expressive power of datalog-like languages for NP search and optimization problems* (Tech. Rep.).
- Gelfond, M., & Lifschitz, V. (1988). The stable model semantics for logic programming. *ICLP*, 1070-1080.
- Gelfond, M., & Lifschitz, V. (1991). Classical negation in logic programs and disjunctive databases. *NGC*, 9(3/4), 365-385.
- Greco, S. (1998). Non-determinism and weak constraints in datalog. *NGC*, 16(4), 373-396.
- Greco, S. (2003). Binding propagation techniques for the optimization of bound disjunctive queries. *IEEE TKDE*, 15(2), 368-385.
- Greco, G., Greco, S., Trubitsyna, I., & Zumpano, E. (2002). Query optimization of disjunctive databases with constraints through binding propagation. *Proceedings of the LPAR Conference* (pp. 216-230).
- Greco, S., Saccà, D., & Zaniolo, C. (1995). The pushdown method to optimize chain logic programs. *Proceedings of the ICALP Conference* (pp. 523-534).

Leone, N., Rullo, P., & Scarcello, F. (1997). Disjunctive stable models: Unfounded sets, fixpoint semantics and computation. *Inf. & Comp.*, 135(2), 69-112.

Lobo, J., Minker, J., & Rajasekar, A. (1992). *Foundations of disjunctive logic programming*. MIT Press.

Minker, J. (1982). On indefinite data bases and the closed world assumption. *Proceedings of the 6th Conference on Automated Deduction* (pp. 292-308).

Ramakrishnan, R., Sagiv, Y., Ullman, J.F., & Vardi, M.Y. (1993). Logical query optimization by proof-tree transformation. *JCSS*, 47(1), 222-248.

Ullman, J.K. (1989). *Principles of database and knowledge-base systems* (vol. 1). Computer Science Press.

## KEY TERMS

**Binding Propagation:** Optimization technique based on the exploitation of binding propagation techniques which reduce the size of the data relevant to answer the query and, consequently, minimize both the complexity of computing a single model and the number of models to be considered.

**Database:** A database (instance) is a set of finite relations over a fixed database schema. Each relation consists of a set of ground facts, that is, variables free facts.

**Disjunctive Datalog Program:** A disjunctive Datalog program is a set of rules  $r$  of the form:

$$A_1 \vee \dots \vee A_k \bullet B_1, \dots, B_m, \neg B_{m+1}, \dots, \neg B_n,$$

where  $k+m+n>0$ ,  $A_1, \dots, A_k, B_1, \dots, B_n$  are atoms of the form  $p(t_1, \dots, t_h)$ ,  $p$  is a predicate symbol of arity  $h$ , and the terms  $t_1, \dots, t_h$  are constants or variables. The disjunction  $A_1 \vee \dots \vee A_k$  is called *head* of  $r$  and is denoted by  $Head(r)$  while the conjunction  $B_1, \dots, B_m, \neg B_{m+1}, \dots, \neg B_n$  is called *body* and is denoted by  $Body(r)$ . If  $k=1$ , then  $r$  is *normal* (i.e.  $\vee$ -free); if  $n=0$ , then  $r$  is *positive* (i.e.  $\neg$ -free); if both  $m=1$  and  $n=0$ , then  $r$  is *normal and positive*; if  $k=n=0$  then  $r$  is a fact, whereas if  $m=0$  then  $r$  is a *constraint* or *denial rule*, that is, a rule which is satisfied only if  $Body(r)$  is false.

**Logic Queries:** A disjunctive datalog *query* over a database defines a mapping from the database to a finite (possibly empty) set of finite (possibly empty) relations

for the goal. A query is a pair  $\langle G, P \rangle$  where  $G$  is an atom, called *goal*, and  $P$  is a program. The application of a query  $Q$  to a database  $D$  is denoted by  $Q(D)$ .

**Magic Set Technique:** Binding propagation technique based on the rewriting of the source query, with respect to the query goal, into an equivalent query which simulates top-down evaluation by bottom-up query computation engines so that irrelevant facts are not considered.

**Query Answer:** The result of a query  $Q = \langle G, P \rangle$  on an input database  $D$  is defined in terms of the stable models of  $P \cup D$  by taking either the union (*possible inference*) or the intersection (*certain inference*) of all models. Thus, given a program  $P$  and a database  $D$ , a ground atom  $G$  is true under possible (brave) semantics if there exists a stable model  $M$  for  $P \cup D$  such that  $G \in M$ . Analogously,  $G$  is true under certain (cautious) semantics if  $G$  is true in every stable model for  $P \cup D$ .

**Query Equivalence:** Two queries  $Q_1 = \langle G_1, P_1 \rangle$  and  $Q_2 = \langle G_2, P_2 \rangle$  are said to be equivalent under semantics  $s$  ( $Q_1 \equiv_s Q_2$ ) if for every database  $D$  (on a fixed schema)  $Ans_s(Q_1, D) = Ans_s(Q_2, D)$ . Two programs  $P_1$  and  $P_2$  are equivalent under a given semantics  $s$ :  $P_1 \equiv_s P_2$  if for every atom  $g$   $\langle g, P_1 \rangle \equiv_s \langle g, P_2 \rangle$ . Finally, if  $Q_1 \equiv_p Q_2$  and  $Q_1 \equiv_c Q_2$  (the two queries or programs are equivalent under both brave and cautious semantics), we simply write  $Q_1 \equiv Q_2$ .

**Stable Models:** An interpretation  $M$  is a stable model for a disjunctive Datalog program  $P$  if and only if  $M$  is a minimal program for the program  $P^M$  denoting the ground positive program derived from  $ground(P)$  by (1) removing all rules that contain a negative literal  $\neg a$  in the body and  $a \in M$ , and (2) removing all negative literals from the remaining rules.

**R**

# A Rhetorical Perspective on Localization and International Outsourcing

**Kirk St. Amant**

*Texas Tech University, USA*

## INTRODUCTION

Globalization is increasingly integrating the economies and the societies of the world. Now, products created in one nation are often marketed to a wide range of international consumers. Similarly, the rapid diffusion of online media has led to an increase in cross-border interactions on both a social and a professional level. Differing cultural expectations, however, can result in miscommunications within this new paradigm of global discourse. Individuals working within this international framework therefore need to understand the process of localization in order to operate more effectively under such a global paradigm. This paper overviews localization, why it is important, and how it might affect professional activities in the future.

## BACKGROUND

To understand localization, one must first understand how rhetoric, or the way in which information is presented, can vary along cultural lines. The idea works as follows: Each culture has its own set of rhetorical expectations, or conditions, for how to convey ideas effectively (Kaplan, 2001; Woolever, 2001). If one presents information in a way that meets the rhetorical expectations of a particular group, then group members will be more inclined to consider that information credible or usable (Bliss, 2001). If one presents information in a way that fails to address a group's rhetorical expectations or that conflicts with those expectations, then the group will view that information as noncredible and will be less inclined to use it. Moreover, if noncredible messages are associated with a particular product, audiences might consider that item as not worth purchasing or using (Ulijn & Strother, 1995).

Just as cultures have different rhetorical expectations, information considered credible and effective by one cultural group might be deemed suspect or unusable by another (Kaplan, 2001; Ulijn & St. Amant, 2000; Woolever, 2001). Language is perhaps the most obvious factor related to credibility in cross-cultural exchanges. If one wishes to develop informative materials for another culture, then concepts must be presented in the language

used by that group. That is, if one wishes to target information for an audience in France, one should use the French language when presenting ideas.

Using the correct language, however, is often not enough, for cultural groups can have different norms for how ideas should be expressed within a language (Bliss, 2001; Driskill, 1996; Kaplan, 2001; Ulijn, 1996). These expectations, moreover, often reflect deep-seated cultural values or societal rules (Ferraro, 2002; Neuliep, 2000). As a result, it can often be difficult for the members of one culture to anticipate the rhetorical expectations another cultural group associates with a credible presentation.

Some cultures, for example, tend to prefer more linear/focused presentations in which facts, connections between ideas, and conclusions are explicitly stated (Campbell, 1998; Ulijn & St. Amant, 2000). Other cultures, however, might prefer more indirect and seemingly circular presentations in which individuals seem to go off on tangents, insert seemingly "random" historical examples, or avoid directly stating facts or conclusions (Campbell, 1998; Ulijn & St. Amant, 2000; Woolever, 2001). These variations can lead to misperceptions or confusion when different cultural groups interact. Ulijn and St. Amant (2000), for example, note that many Western cultures prefer a more direct presentation of information, while many Eastern cultures use a more indirect approach when sharing ideas. As a result, the indirect style used by Eastern cultures is often viewed as "shifty" by Westerners who expect presenters to "get to the point." (In such cultures, a lack of directness equals "dishonesty.") Conversely, many Easterners tend to view the direct presentation style of Western cultures as "rude," for by directly stating information, an individual is patronizing or talking down to the audience. In such cases, by failing to address the rhetorical expectations of the "other" culture, individuals unknowingly undermine their own credibility in cross-cultural exchanges.

Another interesting factor is that cultural rhetorical expectations are not restricted to verbal presentations. Rather, they also affect how different groups perceive and respond to visual displays. As a result, the physical appearance of an object, can differ from country to country. For example, the cultural expectations of what features an item—or visual representations of an item—should

possess can differ from country to country. Such differences can affect how audiences perceive the credibility and the acceptability of visual displays (Kamath, 2000; Lustig & Koester, 1999; Neuliep, 2000).

For example, the perception of a mailbox being a metal or wooden box that sits atop of a post and that has a red flag on the side of it is, essentially, an American one (Gillette, 1999). In other cultures, a mailbox might be a small door in a wall or even a cylindrical metal container that resembles an American fire hydrant. These design discrepancies could then cause confusion when using images to share information across cultures. Consider the following situation: International users come to a Web portal and expect to find a “mail” function. To address this expectation, the portal’s creators have included an “access mail” icon into the portal’s design. The image used for this icon, however, is an American-style mailbox. Unfortunately, this choice of image renders that depiction unrecognizable to users from different cultures—cultures in which mailboxes have very different characteristics. Those individuals might then consider the associated Web portal noncredible, for they perceive it as lacking a key design feature expected of credible Web portals. In this way, cultural differences in visual expectations can affect entire sites that use a particular kind of image.

Moreover, the presence or absence of a single design aspect or feature can be enough to affect the credibility of an image or of an overall Web site. In some cases, cultures can associate different meanings with the same color (Conway & Morrison, 1999; Ferraro, 2002). These associations could thus affect how individuals from different cultures perceive the meaning of a particular image. In the United States, for example, a blue ribbon usually indicates a winner (first place), whereas the same color ribbon in the United Kingdom often indicates second place (in the United Kingdom, a red ribbon is used to signify “first place/winner”). The different associations related to the color blue could affect how Americans vs. Brits perceive a “blue ribbon product” (first rate vs. second rate). In terms of Web design, one might consider how the blue ribbon image associate with the Electronic Frontier Foundation’s (EFF) Blue Ribbon Campaign for Free Speech Online—an icon proudly displayed by many U.S. Web sites—could affect U.K. users’ perceptions of the quality of information found on such sites (second-rate information).

The various expectations cultures can have for visual and verbal communication can markedly affect the success of cross-cultural exchanges. For these reasons, individuals can greatly benefit from practices that address cultural rhetorical differences on both a verbal and a visual level. Localization is a process dedicated to addressing such differences by revising or developing materials in a way that meets the communication expectations of different cultures.

## MAIN THRUST OF THIS ARTICLE

Localization is a process in which professionals design or revise materials to meet the rhetorical expectations of a particular cultural group (Esselink, 2000; Yunker, 2003). Often abbreviated as L10N (for the 10 letters between the “L” and the “N” in “localization”), localization generally involves one of two processes. In both cases, a company or organization wishes to share information with different cultural audiences. The time at which such international sharing takes place, however, affects the tasks of the localizer and the overall process.

In the first scenario, the localization process begins with the creation of original source materials, which are items designed for audiences from a particular cultural group (Esselink, 2000; Yunker, 2003). In such cases, an organization initially creates a product for a particular cultural audience. Both the product and its related documentation are then designed to meet the rhetorical expectations that culture associates with credibility. Over time, the company decides to market the product in other nations. The design of the original item and its related materials might, however, conflict with the rhetorical expectations of those “other” cultural audiences. It is at this point that localizers are used to redesign the product to make it appear credible to users from those other cultures (Esselink, 2000; Yunker, 2003). The task of the localizer then becomes a matter of converting information from the rhetorical styles used in source (original) materials to those of a different cultural audience (often known as the target audience).

In performing such “after-the-fact” conversions, localizers often deal with factors of translation (language) and visual design, in terms of both layout and image use. In such cases, the text is often translated into the language of the desired target audience, and visual and design factors are either revised or replaced in order to match the expectations that same audience (Esselink, 2000; Yunker, 2003). Ideally, this process is a relatively simple “find-and-replace” activity in which items in source materials are replaced with culture specific items for other audiences. Unfortunately, certain factors can affect the ease with which localizers can accomplish such a process.

One of the more interesting problems is text expansion. The idea is that information that can be conveyed with a single word in one language might require multiple words to convey the same meaning in another tongue (Esselink, 2000; Yunker, 2003). The English-language expression “overtime pay” (two words), for example, is often translated as “remuneration des herues supplementaires” (four words) in French. Even in cases in which a single word is used to convey the same concept in different languages, the length of the related word could vary considerably. The English word *help*



(four letters), for example, can be translated into *assistance* (10 letters) in French.

This factor of text expansion is important, for it can affect the design of an overall item. Software products, for example, might use drop-down menus to provide access to different program features. These menus, however, are often configured so that their width accommodates the longest line of text in the original source language. The formatting of a drop-down menu might therefore require redesign to accommodate the text expansion needed to insert corresponding information presented in another language. Moreover, changes resulting from text expansion could require individuals to reconfigure an overall interface to accommodate all of the textual features associated with a program (Esselink, 2000; Yunker, 2003). Similarly, Web site features such as menu buttons or menu options might require redesign to accommodate text expansion. In such cases, the overall size and structure of menu bars or Web pages might also need to be reconfigured to accommodate such changes.

In other instances, the localizer might have to remove or replace certain design features, such as pictures or icons, to address the visual sensibilities of different target cultures (Esselink, 2000; Yunker, 2003). The removal or replacement of such items, however, might create gaps in the overall interface or change the layout of an interface depending on the size of the images that are removed or replaced. As a result, the overall layout of the item might need to be redesigned to accommodate such changes. Additionally, the expectations of a target culture might be so different that creating a localized item involves an entire redesign of an overall product. These factors are only made more complex as after-the-fact localization is often done on relatively tight schedules and with limited budgets. In such cases, localizers often find themselves balancing issues of quality with those of cost and time.

In a second and a more ideal situation, localization would take place from the very beginning of a product's development (Esselink, 2000; Yunker, 2003). In these cases, localizers would do more than simply "revise" source texts created for one specific culture. Rather, localizers would simultaneously generate original source materials for different cultural audiences. Such a process would allow for the simultaneous release of a product into different overseas markets. While such processes seem both time and cost intensive, the quality of the resulting materials would generally be better than items localized after the fact and on a tighter time frame.

Another benefit of simultaneous localization is that localizers might notice design factors, such as the construction of an interface, that would render a particular product less usable (if not unusable) by a certain cultural audience (Esselink, 2000; Yunker, 2003). The localizer

could then present this problem to the designer creating the product. Such consultations allow developers to revise a product during vs. after the initial design process and alleviate many of the problems related to reengineering completed products.

In both scenarios (i.e., after the fact and during development), localization facilitates the flow of information from one cultural group to another by revising materials to meet different rhetorical expectations. While localization practices often focus on products or product related services, localization can be adapted to address processes. Perhaps the most influential reason for such an adaptation is the spread of a production approach known as international outsourcing.

## FUTURE TRENDS

In the past five years, global Internet access has grown dramatically—especially in developing nations. China, for example, has seen its Internet use grow from 2.1 million persons in 1999 to nearly 60 million by the end of 2002 (Section IV Survey Results, 2003; Wired China, 2000). In Africa, the United Nations and private companies have undertaken initiatives to increase online access in specific nations and to the continent in general (Kalia, 2001; Tapping in to Africa, 2000). In Latin America, Global Crossings Ltd. has completed a project that uses fiber optics to give, "multinational companies the ability to communicate with Latin America as efficiently as with any other region" (Tying Latin America Together, 2001, p. 9). And in Eastern Europe, the number of individuals going online is expected to climb from 17% to 27% by 2006 (IDC Research, 2003).

This increased global access has prompted many companies to explore different production methods involving online communication technologies. Perhaps one of the most interesting of these approaches is international outsourcing, a process in which organizations use online media to exchange information and electronic products (e.g., software) with employees located in other countries. The benefit of this process is easy access to labor forces in other countries—namely developing nations with skilled technical workers who can provide services for a fraction of what they would cost in industrialized countries. Online media allow these international workers to exchange information directly and quickly with one another and with the company sponsoring a project. This speed and directness also mean that

- Different parts of an overall process can be performed simultaneously in different locations, and



## A Rhetorical Perspective on Localization and International Outsourcing

- Electronic products (e.g., software) can be forwarded from one location/time zone to another in a manner that would allow work to continue without pause.

Ideally, such processes result in a quality item that can be completed more quickly and at a fraction of the cost it would have taken to produce domestically. These benefits have inspired both private companies and municipal governments to use international outsourcing for everything from computer programming and information technology (IT) work to accounting practices and the staffing of call centers (The New Geography, 2003; Relocating the Back Office, 2003). Moreover, trends in IT and in demographics indicate that offshoring practices will increase in coming years. (For a more detailed discussion of this topic, see Drucker, 2001; Lui & Chan, 2003.)

The key to success in these situations is effective communication. For example, any ignored or misinterpreted information could result in overseas employees performing a process incorrectly. Moreover, the speed with which such processes take place, when combined with the physical and cultural distance separating participants, means that mistakes might go unnoticed until it is too late. Information therefore needs to be conveyed in a way that encourages participants to make use of it. Localization thus becomes an important process, for it increases the chances that outsourcing workers will use essential materials.

Within international outsourcing, localization can take place at two levels. The first level is interface design, and it involves the medium through which individuals interact. In many cases, participants involved in outsourcing come to a central online location (e.g., a company portal or Web page) to collect, present, or exchange information related to a particular product or production process. These interfaces, however, must be designed to encourage use if all involved parties perform their tasks correctly. In these cases, localizers can facilitate the exchange of information by designing interfaces in a way that meets the credibility expectations of different cultures and thus encourages use and information exchange.

The second level of localization application involves how individuals communicate via online media, such as e-mail, chat rooms, and bulletin boards. In these scenarios, participants need to know two important pieces of rhetorical information:

1. How to draft online messages that recipients from other cultural groups will consider credible and worth responding to.
2. How to interpret verbal messages constructed according to different cultural rhetorical norms.

In both cases, localizers can provide participants with two communication “cheat sheets” that address these factors. The first kind of cheat sheet would explain how to draft messages that address the rhetorical expectations of other cultural groups involved in the outsourcing process. The second kind of cheat sheet would tell users how to interpret the meaning that individuals from other cultures convey via a particular rhetorical structure. By teaching others how to address rhetorical factors within the communication process, localizers enhance the chances that participants in international outsourcing activities correctly present and interpret information. Such approaches contribute to the success of projects involving outsourcing.

## CONCLUSION

The global nature of modern business means that individuals must now consider interactions in terms of international audiences. These audiences, however, can have different expectations of how information should be presented in an exchange. Fortunately, localization can facilitate cross-cultural interactions by converting materials from one cultural rhetorical style to another. While localization has historically dealt with products, the shift to international outsourcing allows localization to address processes as well. Thus, by understanding what localization is and how it works, organizations can increase their successes in a variety of cross-cultural communication situations.

## REFERENCES

- Bliss, A. (2001). Rhetorical structures for multilingual and multicultural students. In C. G. Panetta (Ed.), *Contrastive rhetoric revisited and redefined* (pp. 15-30). Mahwah, NJ: Erlbaum.
- Campbell, C. P. (1998). Rhetorical ethos: A bridge between high-context and low-context cultures? In S. Niemeier, C. P. Campbell, & R. Dirven (Eds.), *The cultural context in business communication* (pp. 31-47). Philadelphia: John Benjamin.
- Conway, W. A., & Morrison, T. (1999). The color of money. *Global Business Basics*. Retrieved December 10, 1999, from <http://www.getcustom.com/omnibus/iw0897.html>
- Driskill, L. (1996). Collaborating across national and cultural borders. In D. C. Andrews (Ed.), *International dimensions of technical communication* (pp. 21-44). Arlington, VA: Society for Technical Communication.

- Drucker, P. (2001, November 1). The next society: A survey of the near future. *The Economist*, pp. 3-5.
- Esselink, B. (2000). *A practical guide to localization*. Philadelphia: John Benjamin.
- Ferraro, G. (2002). *Global brains: Knowledge and competencies for the 21<sup>st</sup> century*. Charlotte, NC: Intercultural Associates.
- Gillette, D. (1999, December). Web design for international audiences. *Intercom*, p. 15-17.
- IDC research: Net usage up in Central and Eastern Europe. (2003, Feb. 19). *NUA Internet surveys*. Retrieved June 23, 2003, from [http://www.nua.com/surveys/index.cgi?f=VS&art\\_id=905358723&rel=true](http://www.nua.com/surveys/index.cgi?f=VS&art_id=905358723&rel=true)
- Kalia, K. (2001, July/August). Bridging global digital divides. *Silicon Valley Reporter*, pp. 52-54.
- Kamath, G. R. (2000, May). The India paradox. *Intercom*, pp. 10-11.
- Kaplan, R. B. (2001). Foreword: What in the world is contrastive rhetoric? In C. G. Panetta (Ed.), *Contrastive rhetoric revisited and redefined* (pp. vii-xx). Mahwah, NJ: Erlbaum.
- Lui, K. M., & Chan, K. C. C. (2003). Inexperienced software team and global software team. In A. Gunasekaran, O. Khalil, & S. M. Rahman (Eds.), *Knowledge and information technology management: Human and social perspectives* (pp. 305-323). Hershey, PA: Idea Group.
- Lustig, M., & Koester, J. (1999). *Intercultural competence: Interpersonal communication across cultures* (3<sup>rd</sup> ed.). New York: Longman.
- Neuliep, J. W. (2000). *Intercultural communication: A contextual approach*. Boston: Houghton Mifflin.
- The new geography of the IT industry. (2003, July 17). *The Economist*. Retrieved December 20, 2003, from [http://www.economist.com/displaystory.cfm?story\\_id=S%27%29HH%2EQA%5B%21%23%40%21D%0A](http://www.economist.com/displaystory.cfm?story_id=S%27%29HH%2EQA%5B%21%23%40%21D%0A)
- Relocating the back office. (2003, December 11). *The Economist*. Retrieved December 20, 2003, from [http://www.economist.com/displaystory.cfm?story\\_id=2282381](http://www.economist.com/displaystory.cfm?story_id=2282381)
- Section IV survey results. (2003, January). *Semiannual report on the development of China's Internet*. Retrieved June 25, 2003, from <http://www.cnnic.net.cn/develst/2003-1e/444.shtml>
- Tapping in to Africa. (2000, September 9-15). *The Economist*, p. 49.
- Tying Latin America together. (2001, Summer). *NYSE Magazine*, p. 9.
- Ulijn, J. M. (1996). Translating the culture of technical documents: Some experimental evidence. In D. C. Andrews (Ed.), *International dimensions of technical communication* (pp. 69-86). Arlington, VA: Society for Technical Communication.
- Ulijn, J. M., & St. Amant, K. (2000). Mutual intercultural perception: How does it affect technical communication—Some data from China, the Netherlands, Germany, France, and Italy. *Technical Communication*, 47 (2), 220-237.
- Ulijn, J. M., & Strother, J. B. (1995). *Communicating in business and technology: From psycholinguistic theory to international practice*. Frankfurt, Germany: Peter Lang.
- Wired China. (2000, July 22). *The Economist*, pp. 24-28.
- Woolever, K. R. (2001). Doing global business in the information age: Rhetorical contrasts in the business and technical professions. In C. G. Panetta (Ed.), *Contrastive rhetoric revisited and redefined* (pp. 47-64). Mahwah, NJ: Erlbaum.
- Yunker, J. (2003). *Beyond borders: Web globalization strategies*. Boston: New Riders.

## KEY TERMS

**International Outsourcing:** Production process in which individuals in other nations perform work for an organization.

**Localization:** The process of revising materials designed for one particular culture to meet the communication expectations of a different cultural group.

**Online Media:** Electronic communication technologies that rely on the Internet or the World Wide Web as a mechanism for presenting or exchanging information.

**Rhetoric:** The manner in which information is presented.

**Source–Source Text–Source Materials:** Original materials designed for a specific cultural group.

**Target Culture:** The culture for which one revises source materials in the localization process.

**Text Expansion:** Situation in which one language requires more words to express the same meaning than does another language.

# Rough Sets

**Zdzislaw Pawlak**

*Polish Academy of Sciences and Warsaw School of Information Technology, Poland*

**Lech Polkowski**

*Polish-Japanese Institute of Information Technology and University of Warmia and Mazury, Poland*

**Andrzej Skowron**

*Warsaw University, Poland*

## INTRODUCTION

Rough set theory is a new mathematical approach to imperfect knowledge. The problem of imperfect knowledge, tackled for a long time by philosophers, logicians, and mathematicians, has become also a crucial issue for computer scientists, particularly in the area of artificial intelligence. There are many approaches to the problem of how to understand and manipulate imperfect knowledge. The most successful one is, no doubt, fuzzy set theory proposed by Zadeh (1965). Rough set theory (Pawlak, 1982) presents still another attempt at this problem. This theory has attracted the attention of many researchers and practitioners all over the world, who contributed essentially to its development and applications. The rough set approach seems to be of fundamental importance to AI and cognitive sciences, especially in the areas of machine learning, knowledge acquisition, decision analysis, knowledge discovery from databases, expert systems, inductive reasoning, and pattern recognition.

The rough set approach provides efficient algorithms for finding hidden patterns in data, minimal sets of data (data reduction), evaluating the significance of data, and generating sets of decision rules from data. This approach is easy to understand and offers straightforward interpretation of obtained results, and most of its algorithms are particularly suited for parallel processing.

The rough set philosophy is founded on the assumption that with every object of the universe of discourse we associate some information. For example, if objects are patients suffering from a certain disease, symptoms of the disease form information about patients. Objects characterized by the same information are indiscernible (similar) in view of the available information about them. The *indiscernibility relation* generated in this way is the mathematical basis of rough set theory.

Any set of all indiscernible (similar) objects is called an *elementary set (neighborhood)* and forms a basic *granule (atom)* of knowledge about the universe. Any

union of elementary sets is referred to as a *crisp (precise) set*; otherwise, the set is *rough (imprecise, vague)*.

Consequently each rough set has *boundary-line cases*, i.e., objects which cannot with certainty be classified either as members of the set or of its complement. Obviously crisp sets have no boundary-line elements at all. This means that boundary-line cases cannot be properly classified by employing the available knowledge.

Vague concepts, in contrast to precise concepts, cannot be characterized in terms of information about their elements. Therefore, in the proposed approach, we assume that any vague concept is replaced by a pair of precise concepts, called the lower and the upper approximation of the vague concept. The lower approximation consists of all objects which surely belong to the concept, and the upper approximation contains all objects which possibly belong to the concept. The difference between the upper and the lower approximation constitutes *the boundary region* of the vague concept. Approximations are two basic operations in rough set theory. The observation that vague concepts should have a nonempty boundary was made by Gottlob Frege in the beginning of 20th century.

## BACKGROUND

In Table 1, we consider essential issues in rough sets in more detail.

## ROUGH SETS AS A TOOL FOR REASONING ABOUT VAGUE CONCEPTS

Vague complex concept approximation and reasoning about vague concepts by means of such approximations become critical for numerous applications related to multiagent systems (such as Web mining, e-commerce,

Table 1. A summary of basic research ideas of rough set theory

**Data tables.** Data for rough set based analysis are usually formatted into a data table (an *information system*)  $A = (U, A)$ , where the set  $U$  consists of *objects* (e.g., records, signals, processes, patients) and the set  $A$  consists of *attributes* (e.g., physical parameters, features expressed in symbolic or numerical form, results of medical tests); any attribute  $a \in A$  is a mapping from  $U$  into a value set  $V_a$ . Subsets of  $U$  are *concepts*.

**Indiscernibility relation.** The *A-indiscernibility relation*,  $IND(A)$  is defined as follows,  $x IND(A) y$  iff  $a(x)=a(y)$  for  $a \in A$ . For  $B \subseteq A$ , one may consider a restricted information system  $A(B) = (U, B)$  and define the *B-indiscernibility*  $IND(B)$ . For a set  $B$  of attributes, we denote by  $[x]_B$  the equivalence class of  $x \in U$  with respect to  $IND(B)$ . In terms of indiscernibility, important notions related to knowledge reduction and attribute dependence are expressed.

**Reducts.** For a set  $B$  of attributes, one can look after an inclusion-minimal set  $C \subseteq B$  with the property that  $IND(C)=IND(B)$ , i.e.,  $C$  is a minimal subset of attributes in  $B$  that provides the same classification of concepts as  $B$ . Such  $C$  is said to be a *B-reduct*. The problem of finding a minimum-length reduct is NP-hard (Skowron & Rauszer, 1992), so heuristics are used in searching for short (or relevant with respect to a give criterion) reducts. We mention heuristics based on the Johnson algorithm or genetic algorithms (Bazan et al, 1998). Given a  $B$ -reduct  $C$ , one can reduce the information system  $A(B)$  to the system  $A(C)$  without any loss of classification ability. Many other kinds of reducts and their approximations are discussed in literature. They are used in searching for relevant patterns in data (Polkowski & Skowron, 1998; Polkowski, Tsumoto & Lin, 2002).

**Functional dependence.** For given  $A = (U, A)$ ,  $C, D \subseteq A$ , by  $C \rightarrow D$  is denoted the *functional dependence* of  $D$  on  $C$  in  $A$  that holds iff  $IND(C) \subseteq IND(D)$ . In particular, any  $B$ -reduct  $C$  determines functionally  $D$ . Also dependencies to a degree are considered (Pawlak, 1991).

**Definable and rough concepts (sets).** Classes of the form  $[x]_B$  can be regarded as the primitive *B-definable concepts* whose elements are classified with certainty by means of attributes in  $B$ . This property extends to more general concepts, i.e., a concept  $X \subseteq U$ , is *B-definable* iff for each  $y$  in  $U$ , either  $[y]_B \subseteq X$  or  $[y]_B \cap X = \emptyset$ . This implies that  $X$  has to be the union of a collection of  $B$ -indiscernibility classes, i.e.,  $X = \cup \{[x]_B : x \in X\}$ . Then we call  $X$  a *B-exact (crisp, precise) concept*. One observes that unions, intersections and complements in  $U$  to  $B$ -exact concepts are  $B$ -exact as well, i.e.,  $B$ -exact concepts form a Boolean algebra for each  $B \subseteq A$ . In case when a concept  $X$  is not  $B$ -exact, it is called *B-rough*, and then  $X$  is described by *approximations* of  $X$  that are exact concepts (Pawlak, 1991), i.e., one defines the *B-lower approximation* of  $X$ , and the *B-upper approximation* of  $X$  by  $B_*(X) = \{x \in X : [x]_B \subseteq X\}$  and  $B^*(X) = \{x \in X : [x]_B \cap X \neq \emptyset\}$ , respectively. The set  $B^*(X) - B_*(X)$  is called the *B-boundary region* of  $X$ .

**Rough membership functions.** Roughness of  $Y$  with respect to a set  $B$  can be measured by some coefficients (Pawlak, 1991). A precise local characteristics of a concept  $X$  with respect to a classification  $IND(B)$  in an information system  $(U, A)$  can be given by a *rough membership function* (Pawlak & Skowron, 1994), defined as the fraction of the class  $[x]_B$  contained in  $X$ . This definition is relative to a given source of information what makes the rough membership function different from fuzzy membership function. Notice that such a coefficient has been considered by Lukasiewicz (:ukasiewicz, 1913) long time ago in studies on assigning fractional truth values to logical formulas.

**Decision systems and rules.** Matching classification of objects by an expert with a classification in terms of accessible features, can be done with decision systems. A *decision system* is a tuple  $A^d = (U, A, d)$ , where  $(U, A)$  is an information system with the set  $A$  of condition attributes, and the decision (attribute)  $d: U \rightarrow V_d$ , where  $d \notin A$ . In case  $A \rightarrow d$  holds in  $A^d$ , we say that the decision system  $A^d$  is deterministic and the dependency  $A \rightarrow d$  is  $A^d$ -exact. Then, for each class  $[x]_A$  there exists a unique decision  $d(x)$  throughout the class. Otherwise, the dependency  $A \rightarrow d$  in  $A^d$  holds to a degree. A *decision rule* in  $A^d$  is any expression  $\wedge \{a=v_a : a \in A \text{ and } v_a \in V_a\} \rightarrow d=v$  where  $d$  is the decision attribute and  $v \in V_d$ . This decision rule is true in  $(U, A, d)$  if for any object satisfying its left hand side it also satisfies the right hand side, otherwise the decision rule is true to a degree measured by some coefficients (Pawlak, 1991). Strategies for inducing decision rules can be found in (Polkowski & Skowron, 1998; Polkowski, Tsumoto & Lin, 2000).



Table 1. A summary of basic research ideas of rough set theory, cont.

<p><b>Approximation spaces.</b> Several generalizations of the classical rough set approach based on approximation spaces have been reported in the literature. Let us consider some examples. A <i>generalized approximation space</i> is defined by a tuple <math>AS=(U,I,v)</math> where <math>I</math> is the <i>uncertainty function</i> defined on <math>U</math> with values in the powerset <math>P(U)</math> of <math>U</math> (<math>I(x)</math> is the neighborhood of <math>x</math>) and <math>v</math> is the <i>inclusion function</i> (called also <i>rough inclusion</i>) defined on the Cartesian product <math>P(U) \times P(U)</math> with values in the interval <math>[0,1]</math> measuring the degree of inclusion of sets. The standard rough inclusion is defined by <math>v(X,Y) =  X \cap Y / X </math> if <math>X</math> is nonempty, and 1 otherwise, for <math>X, Y \subseteq U</math>. The lower <math>AS_*</math> and upper <math>AS^*</math> approximation operations can be defined in <math>AS</math> by <math>AS_*(X) = \{x \in U: v(I(x), X) = 1\}</math> and <math>AS^*(X) = \{x \in U: v(I(x), X) &gt; 0\}</math>, respectively. In the standard case <math>I(x)</math> is equal to the equivalence class <math>[x]_B</math> of the indiscernibility relation <math>IND(B)</math>; in case of tolerance (similarity) relation <math>\tau \subseteq U \times U</math> we take <math>I(x) = \{y \in U: x\tau y\}</math>, i.e., <math>I(x)</math> is equal to the tolerance class of <math>\tau</math> defined by <math>x</math>. Usually, there are considered families of approximation spaces with approximation spaces labeled by some parameters. By tuning such parameters according to chosen criteria (e.g., the minimal description length principle) one can search for the optimal approximation space for concept description.</p> <p><b>Rough mereology.</b> The approach based on inclusion functions was generalized to the <i>rough mereological approach</i> (Polkowski &amp; Skowron 1996; Polkowski, Tsumoto &amp; Lin, 2000; Pal, Polkowski &amp; Skowron, 2004). The inclusion relation <math>x \mu_r y</math> with the intended meaning “<math>x</math> is a part of <math>y</math> to a degree at least <math>r</math>” has been taken as the basic notion of the <i>rough mereology</i> that is a generalization of the Leśniewski mereology. Rough mereology offers a methodology for synthesis and analysis of complex objects in distributed environment of intelligent agents, in particular, for synthesis of objects satisfying a given specification to a satisfactory degree or for control in such complex environment. Moreover, rough mereology has been recently used for developing foundations of the <i>information granule calculi</i> (Pal, Polkowski &amp; Skowron, 2004), aiming at formalization of the Computing with Words and Perceptions paradigm, recently formulated in (Zadeh, 2001). More complex information granules are defined recursively using already defined information granules and their measures of <i>inclusion</i> and <i>closeness</i>. Information granules such as classifiers (Kloesgen &amp; Zytkow, 2002) or approximation spaces can have complex structures. Computations on information granules are performed to discover relevant information granules, e.g., patterns or approximation spaces for complex concept approximations. There are also developed extensions of rough sets to deal with concept approximations using inductive reasoning.</p>
---

monitoring, security and rescue tasks in multiagent systems, cooperative problem solving, intelligent smart sensor fusion, human-computer interfaces, telemedicine, and soft views of databases for specific customers). Further development of such methods for approximate reasoning by intelligent agents (Russell & Norvig, 2003) based on databases and knowledge bases is needed.

Tasks collected under the labels of data mining, knowledge discovery, decision support, pattern classification, and approximate reasoning require tools aimed at discovering templates (patterns) in data and classifying them into certain decision classes. Templates are in many cases the most frequent sequences of events, most probable events, regular configurations of objects, the decision rules of high quality, and approximate reasoning schemes. Tools for discovering and classifying templates are based on reasoning schemes rooted in various paradigms (Kloesgen & Zytkow, 2002). Such patterns can be extracted from data by means of methods based on Boolean reasoning and discernibility. The *discernibility relation* (in the simplest case defined as the complement of the indiscernibility relation) is one of the most important relations considered in rough set theory. The ability to discern between per-

ceived objects is important for constructing many entities like reducts, decision rules, or decision algorithms. The idea of *Boolean reasoning* is based on construction for a given problem  $P$  of a corresponding Boolean function  $f_p$  with the following property: The solutions for the problem  $P$  can be decoded from prime implicants of the Boolean function  $f_p$ . Let us mention that to solve real-life problems it is necessary to deal with Boolean functions having a large number of variables.

A list of some current research directions on the rough set foundations and the rough-set-based methods is presented in Table 2.

For more details the reader is referred to the enclosed bibliography on rough sets (Demri & Orowska, 2002; Lin & Cercone, 1997; Pal et al., 2004; Pal & Skowron, 1999; Pawlak, 1991; Polkowski, 2002; Polkowski & Skowron, 1998; Polkowski et al., 2000).

## FUTURE PLANS

To enhance the above-mentioned methods (see Table 3), further development of methods based on rough sets in



Table 2. A list of research directions on rough sets

<ul style="list-style-type: none"> <li>• Boolean reasoning and approximate Boolean reasoning strategies as the basis for efficient heuristics for rough set methods.</li> <li>• Tolerance-(similarity)-based rough set approach.</li> <li>• Rough-set-based approach based on neighborhood (uncertainty) functions and inclusion relation. In particular, variable precision rough set model.</li> <li>• Rough sets in multi-criteria decision analysis and preference modeling.</li> <li>• Rough sets and non-deterministic information systems.</li> <li>• Rough-set-based clustering.</li> <li>• Rough sets and incomplete information systems. In particular, missing value problems.</li> <li>• Rough sets and noisy data.</li> <li>• Rough sets and relational databases.</li> <li>• Rough sets and inductive reasoning.</li> <li>• Rough sets in modeling of decision systems and analysis of complex systems, in particular, rough sets and layered (hierarchical) learning.</li> <li>• Rough sets as a tool for approximate reasoning in distributed systems, by autonomous agents, and in multiagent systems.</li> <li>• Calculi of information granules.</li> </ul>
--

Table 3. An assessment of results and challenges

<ul style="list-style-type: none"> <li>• A successful methodology based on the discernibility of objects and Boolean reasoning was developed for computing of many different kinds of reducts and their approximations that are used (i) for inducing, e.g., decision rules, association rules, discretization of real value attributes, symbolic value grouping, (ii) in searching for new features defined by oblique hyperplanes, higher order surfaces, and relevant patterns from data, (iii) in conflict resolution or negotiation.</li> <li>• Most of the problems related to generation of the above reducts are NP-complete or NP-hard. However, it was possible to develop efficient heuristics returning suboptimal solutions of the problems. The results of experiments on many data sets are very promising. They show very good quality of solutions generated by the heuristics in comparison with other methods reported in literature (e.g., with respect to the classification quality of unseen objects). Moreover, they are very efficient from the point of view of time necessary for computing of the solution. It is important to note that the methodology makes it possible to construct heuristics having a very important approximation property: expressions generated by heuristics (i.e., implicants) close to prime implicants define approximate solutions for the problem (Polkowski &amp; Skowron, 1998).</li> <li>• A wide range of applications of methods based on rough set theory alone or in combination with other approaches have been discovered in the following areas (Polkowski &amp; Skowron, 1998; Pal, Polkowski &amp; Skowron, 2004): acoustics, biology, business and finance, chemistry, computer engineering (e.g., data compression, digital image processing, digital signal processing, parallel and distributed computer systems, sensor fusion, fractal engineering), decision analysis and systems, economics, electrical engineering (e.g., control, signal analysis, power systems), environmental studies, digital image processing, informatics, medicine, molecular biology, musicology, neurology, robotics, social science, software engineering, spatial visualization, Web engineering, and Web mining.</li> <li>• Several software systems based on rough sets have been developed such as RSES (see references).</li> <li>• Rough sets in combination with other soft and computing technologies such as fuzzy sets, evolutionary programming, neural networks or crisp technologies offered, e.g., by statistical or analytical techniques are promising in solving some tasks however they should be further developed to deal with hard real-life problems.</li> </ul>
---

combination with other soft computing techniques is needed. In particular, methods based on rough mereological approach (Polkowski & Skowron, 1996) combined with information granulation (Pal et al., 2004) seems to be very promising. Among issues to be investigated are scalability problems, methods for designing adaptive systems, and methods for constructing intelligent sys-

tems that perform computations with words and perceptions (Zadeh, 2001).

## CONCLUSION

Rough set theory supplies essential tools for knowledge analysis. It allows for creating algorithms for knowledge

reduction, concept approximation, decision rule induction, and object classification. The methods of rough set theory rest on indiscernibility and related notions, in particular, on notions related to rough inclusions. All constructs needed in implementing rough-set-based algorithms can be derived from data tables with need for neither a priori estimates nor preliminary assumptions. Currently, research in rough set theory is directed, among others, at problems of knowledge granulation, techniques for computing with words and perceptions, and rough-neural computing (Pal et al., 2004).

## REFERENCES

- Bazan, J., Nguyen, H. S., Nguyen, S. H., Synak, P., & Wróblewski, J. (2000). Rough set algorithms in classification problems. In L. Polkowski, S. Tsumoto, & T. Y. Lin (Eds.), *Rough set methods and applications: New developments in knowledge discovery in information systems* (pp. 49-88). Heidelberg, Germany: Physica-Verlag.
- Demri, S., & Orłowska, E. (2002). *Incomplete information: Structure, inference, complexity*. Heidelberg, Germany: Springer-Verlag.
- Kloesgen, W., & Zytkow, J. (Eds.). (2002). *Handbook of knowledge discovery and data mining*. Oxford, UK: Oxford University Press.
- Lin, T. Y., & Cercone, N. (1997). *Rough sets and data mining: Analysis of imperfect data*. Boston: Kluwer Academic.
- Lukasiewicz, J. (1970). Die logischen Grundlagen der Wahrscheinlichkeitsrechnung (1913). In L. Borkowski (Ed.), *Jan Lukasiewicz: Selected works*, Amsterdam; Warsaw, Poland: Polish Scientific Publishers and North-Holland
- Pal, S. K., Polkowski, L., & Skowron, A. (Eds.). (2004). *Rough-neural computing: Techniques for computing with words*. Berlin, Germany: Springer-Verlag.
- Pal, S. K., & Skowron, A. (1999). *Rough fuzzy hybridization: A new trend in decision-making*. Singapore: Springer-Verlag.
- Pawlak, Z. (1982). Rough sets, *International Journal of Computer and Information Sciences*, 11, 341-356.
- Pawlak, Z. (1991). *Rough sets: Theoretical aspects of reasoning about data*. Dordrecht, Germany: Kluwer.
- Pawlak, Z., & Skowron, A. (1994). Rough membership functions. In R. R. Yager, M. Fedrizzi, & J. Kacprzyk (Eds.), *Advances in the Dempster-Schafer theory of evidence* (pp. 251-271). New York: Wiley.
- Polkowski, L. (2002). *Rough sets: Mathematical foundations*. Heidelberg, Germany: Physica-Verlag.
- Polkowski, L., Tsumoto, S., & Lin, T. Y. (Eds.). (2000). *Rough set methods and applications: New developments in knowledge discovery in information systems*. Heidelberg, Germany: Physica-Verlag.
- Polkowski, L., & Skowron, A. (1996). Rough mereology: A new paradigm for approximate reasoning., *International Journal of Approximate Reasoning*, 15(4), 333-365.
- Polkowski, L., & Skowron, A. (Eds.). (1998). *Rough sets in knowledge discovery 1 & 2*. Heidelberg, Germany: Physica-Verlag.
- Russell, S. J., & Norvig, P. (2003). *Artificial intelligence. A modern approach*. Saddle River, NJ: Prentice Hall.
- Skowron, A., & Rauszer, C. (1992). The discernibility matrices and functions in information systems. In R. Slowiński (Ed.), *Intelligent decision support: Handbook of applications and advances of the rough sets theory* (pp. 331-362). Dordrecht, Germany: Kluwer.
- Skowron, A., & Stepaniuk, J. (2001). Information granules: Towards foundations of granular computing. *International Journal of Intelligent Systems*, 16(1), 57-86.
- Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8, 338-353.
- Zadeh, L. A. (2001). A new direction in AI: Toward a computational theory of perceptions. *AI Magazine*, 22(1), 73-84.

## KEY TERMS

**Boolean Reasoning:** Is based on construction for a given problem P of a corresponding Boolean function  $f_p$  with the following property: The solutions for the problem P can be decoded from prime implicants of the Boolean function  $f_p$ .

**Decision Rule:** In  $(U, A, d)$  is any expression of the form  $\bigwedge \{a=v_a : a \in A \text{ and } v_a \in V_a\} \rightarrow d=v$  where  $d$  is the decision attribute and  $v$  is a decision value. This decision rule is true in  $(U, A, d)$  if for any object satisfying its left-hand side it also satisfies the right-hand side; otherwise, the decision rule is true to a degree measured by some coefficients such as confidence.

**Decision System:** Is a tuple  $(U, A, d)$ , where  $(U, A)$  is an information system with the set  $A$  of condition attributes and the decision (attribute)  $d: U \rightarrow V_d$ , where  $d \notin A$ . Informally,  $d$  is an attribute whose value is given by an external source (oracle, expert) in contradiction to conditional attributes in  $A$  whose values are determined by the user of the system.

**Functional Dependence:** For attribute sets  $C, D$ , we say that  $D$  depends functionally on  $C$ , in symbols  $C \rightarrow D$ , in case  $IND(C) \subseteq IND(D)$ . Also non-exact (partial) functional dependencies to a degree are considered.

**Indiscernibility Relation:** Is defined as follows: Objects  $x, y$  are indiscernible iff information about  $x$  is equal to (similar with) information about  $y$ . In the former case the indiscernibility relation is an equivalence relation; in the latter it is a similarity relation. Any object  $x$  defines an indiscernibility class (neighborhood) of objects indiscernible with this object. Also soft cases of indiscernibility relation are considered. Discernibility relation is a binary relation on objects defined as follows: Objects  $x, y$  are discernible iff information about  $x$  is discernible from information about  $y$ . In the simplest case, objects  $x, y$  are discernible iff it is not true that they are indiscernible.

**Information About Object:**  $x$  in a given information system  $(U, A)$  is defined by  $Inf_A(x) = \{(a, a(x)) : a \in A\}$ .

**Information System:** Is a pair  $(U, A)$  where  $U$  is the universe of objects and  $A$  is a set of attributes, i.e., functions on  $U$  with values in respective value sets  $V_a$  for  $a \in A$ .

**Lower (Upper) Approximation:** Of a set  $X \subseteq U$  is the union of all indiscernibility classes contained in  $X$  (that intersect  $X$ ), i.e., it is the greatest exact set contained in  $X$  (the smallest exact set containing  $X$ ). Instead of exact containment also containment to a degree is used.

**Reduct:** Is a minimal with respect to inclusion subset  $C$  of  $B$  preserving a given indiscernibility (discernibility) constraint, e.g.,  $IND(C) = IND(B)$ . Many different kinds of reducts with respect to different discernibility criteria have been investigated and used in searching for relevant patterns in data.

**Rough Set:** Is a subset (concept) of the universe of objects  $U$  in an information system  $(U, A)$  that cannot be expressed (defined) as a union of indiscernibility classes; otherwise, the set is called **exact**.

# Security Controls for Database Technology and Applications

**Zoltán Kincses**

*Eötvös Loránd University of Sciences, Hungary*

## INTRODUCTION

The word *security* has many meanings, even for the programmers and other members of the computer science community, but it is often inferred by frequently used keywords, such as *threat*, *vulnerability*, *exploit*, *risk*, and related expressions such as *attack*, *circumvention*, *manipulation*, *sniffing*, *denial of service*, *malicious*, *privacy*, and so forth.

Every system is at risk of being attacked, and the risk analysis of the system will discover the severity of these risks, the protection measures, and their efficiency. There may be situations in which the risk analysis concludes that the system is not well protected against an attack, or even that new risks arose during a periodical analysis. The important thing is to implement the analysis and, based on the results, decide how to proceed with the next steps.

Without the risk analysis there is no guarantee that all the protections will be sufficient, and the owner or maintainer of the system will be consciously regarding the security of the system. We can conclude, therefore, that *security is conscious risk taking*.

## WHAT IS INFORMATION SECURITY?

Information security can be defined in different ways, depending on the starting point. The word *security* has different meanings for society, for the government, for a citizen, for a mother, for a company leader, for a developer, for a user, or even for a hardware object. The specific rules for information security can be learnt from good books and descriptions found on the Internet. The “security way of thinking,” however, can be learnt from a fewer number of books, publications, and newsletters (see, e.g., Anderson, 2001; Schneier, 1996, 2000, 2003). Beginners should, and professionals may, read these papers before editing configuration files, using security applications, and applying security protocols, procedures, and policies to their systems.

Information security is realized through settings and installations, but without built-in concepts of the security policies (i.e., concepts built in from the beginning,

in the planning state of the system), information security has less value and, in many cases, is worthless because of the bad concept. As Schneier explained, “security is a process, not a product.” But he also says that “security is a trade-off”, therefore we must be careful with applied security measures.

Security is a conscious risk taking, and, based on this idea, security means that the manager (user, developer, etc.) must consciously manage the risk of the system. Risk can be managed with the cycle of monitoring (MO), planning and organization (PO), acquisition and implementation (AI), and delivery and support (DS) processes defined by the COBIT system (Information Systems Audit and Control Association [ISACA], 2003). In this system the control measures can be placed, which will help us find possible security problems.

The relations among the security parameters (technical and managerial) and their effect on each other is shown in Figure 1.

## MAIN THRUST OF THE CHAPTER

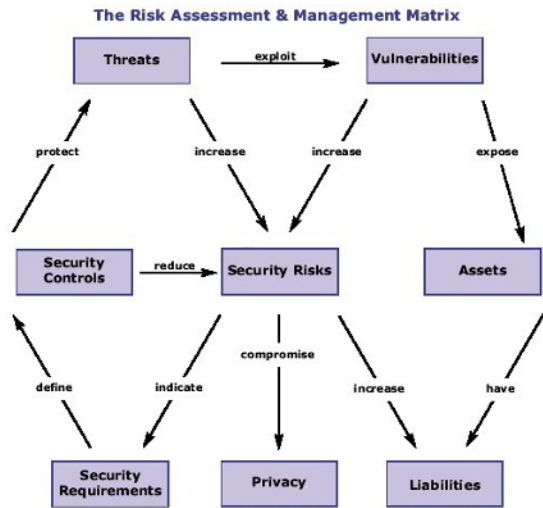
Controls will help us find possible security problems in a system. The best controls are *preventive*, followed by *detective* controls. At the least, *corrective* controls must be provided, because the business continuity is served in that way (Stoneburner, Goguen, & Feringa, 2002).

Additional controls exist, such as administrative controls, but this article will detail only the three previously introduced. Administrative controls have a strong relation to the quality assurance system (i.e., ISO9001 series), because the existence, the content, and the form of policies, documentations, and instructions are regulated by that standard. After introducing such a system, it is easier to complete the system with the security-related parts.

## Samples and Rules

A sample of controls, and their types, are picked from different areas of everyday life, and include locks (preventive control), house alarms (detective control), and

Figure 1. Security map. Note. From IT Security: Hackers, Crackers, and Thugs, by NCR, 2002. Available online from [http://www.ncr.com/repository/articles/pdf/wcs\\_security.pdf](http://www.ncr.com/repository/articles/pdf/wcs_security.pdf)



insurance (corrective control). Samples from IT systems include firewalls (preventive control), intrusion detection systems (detective controls), and backup systems (corrective control).

Which security tool or defence technique can resolve certain controls, and which security tool or defence technique implements certain control or controls? The answer to these two questions must produce the same result at the end of the system's security planning. If we consider the two sets of controls (C) and tools (T), then the C→T relation must fulfil the following rules:

1.  $\forall t \in T, \exists c \in C: C(c) \rightarrow T(t)$  and
2.  $\forall c \in C \exists t \in T: C(c) \rightarrow T(t)$

The first rule means that in set of tools there are no unused tools, and the second rule means that in set of controls there are no unsupported controls by at least one tool. In many-to-many relations, the segregation of duties must be applied, which means the avoidance of overlapping. It is not very efficient to use two virus scanners on a computer or think that every control can be supported by one universal tool. Of course, well-considered cases can be suitable (e.g., one firewall can stop e-mails that contain viruses or can have intrusion-detection system features).

Multiple controls and separate tools can be applied on different levels (e.g., a virus scanner can be on a

firewall and on a single computer, also). The borders of the levels must be defined in advance, and every change in the borders or in the level membership of the computer must be followed by an audit, the topic of which is the consistency of the system and its security. The result may be that the change cannot be done in that way or that additional controls must be applied to ensure consistency.

## Threats

The control measures concern the basic threats to the system. There are three basic threats: *confidentiality* (C), *integrity* (I), and *availability* (A).

*Confidentiality* means that unauthorized users or processes cannot access the protected information (e.g., data, application, configuration settings, operating system, network). Considering the information life cycle, from its appearance till the unrecoverable removal, confidentiality exists only till the first unauthorized access. The accesses can be controlled by authentication and authorization (preventive), or by access alerter and monitoring the access logs (detective). The corrective control also could start with the monitoring of access logs and, in case of unauthorized access, the cause must be eliminated. Unfortunately, many or all parts of the information system might become nonconfidential, that is, only the new entries after the elimination of the unauthorized access possibility can be treated as confidential.

Confidentiality generally can be resolved with encryption techniques. Symmetric encryption ensures that only those users who possess the key for the access can access the information. If it is necessary for only one user to access the information, then an asymmetric or public key encryption system can be used. In the case of managing many users and public keys, the public key infrastructure (PKI) is used.

For database technologies and applications, this means that in case of unauthorized access or manipulation, the manipulated data must be replaced with the original data. In case of unauthorized access, the data confidentiality cannot be restored, but the confidentiality of new entries can be resolved by replacing the insufficient confidentiality control with a sufficient one.

*Integrity* means that the information was not modified in part or in whole. The integrity can be checked with the backup copy, but there are also encryption techniques for this problem. The hash algorithms with one-way function methods produce a standard output from any input, and the probability of having the same the output values from two inputs is very low. This short



output (e.g., 20 bytes) is good for large-size information also, and it can be easily checked that the backup and the live data have the same hash.

Integrity also can be ensured by digital signature, in which the sender or data producer signs the data and, in case of manipulation (even by just one byte), the receiver will recognize that the signatures are not matching. (For details about cryptography, see Schneier, 1996.) In a PKI system, the public key of the receiver is for confidentiality (only the owner of the secret key can decrypt the information), whereas the secret key of the sender is for integrity (the signature can be done only by the secret key of the sender, and everybody can check it with the public key of the sender).

For database technologies and applications, this means that in case of data manipulation or partially compromised data, the intact database or partial data must be reloaded.

*Availability* means that the resource is available. The resource can be a physical device or information, and its availability must be assured for authorized persons with good intentions. Bad intentioned and unauthorized persons are considered attackers. In some cases also the authorized users can damage the availability by extreme usage of the resource. This causes the system to be unavailable, and in the case of an intentional behaviour this is called a denial of service (DoS) attack. When many computers are involved in extreme resource demand, then the attack is called distributed DoS.

Availability can be ensured by reserve systems, called backup systems, where damaged resources can be replaced or quickly recovered. Good backups for information stored and transactions processed in the system must have a good backup policy. Not only are timely backups necessary, but the tests that show if these backups can be used in a case of emergency are also necessary. For database technologies and applications, this means that, in case of data damage or loss, the main value must be restored as soon as possible from at least the previous backup.

The basic threats and the controls form a preventive-detective-corrective/confidentiality-integrity-availability (PreDeCo/CIA; ISACA, 2003) table, which can be filled by the project manager with the help of developers. Together can be found the possible security threats in the system.

### PreDeCo/CIA Table

In the PreDeCo/CIA table, each cell is detailed, and the table contains only the briefest features of the cell. The questions are specified depending on the area where this method is applied. There are nine cells:

- **Preventive confidentiality:** The main threat is that somebody will access the system or a part of it without legal permission, because of bad access-rights settings or by insufficient protection of confidential information. For example, a guest user can read the files of other users. In this case, adequate settings of the access right and the encryption may help. Even if the user is able to access the file, the content will remain confidential because of the encryption.
- **Preventive integrity:** The attacker can change the information or the system settings. The situation is worst case if there is no control to check–detect–correct this manipulation. Hashes should be applied to database integrity, depending on the size of the database either on the entire table or on subparts (e.g., rows, columns, cells).
- **Preventive availability:** The attacker can reach that the database is not available. The multiplied service resources can prevent this, where the balance can be set dynamically depending on the load.
- **Detective confidentiality:** In the worst case, top-secret data will be compromised and disclosed to unauthorized competitors. Access detection controls could help, but if they detect the action after it occurs, it is too late. For example, an attacker could conduct an analysis on a database created for a survey and profit from the results. One protection is making sure the data cannot be read or used, even in case of possession. Some encryption techniques will serve in this case.
- **Detective integrity:** Same as in case of preventive control. There are integrity hashes and here are the checkers of these hashes. If the checkers detect integrity manipulation of the system parameters, they must react immediately.
- **Detective availability:** A DoS or DDoS attacks the system or a part of it, but sometimes the partial attack can cause the unavailability of the whole system. For example, the attacker targets the Web server and the whole operating system crashes.
- **Corrective confidentiality:** If the threats in the previous paragraphs happened, the data are no longer confidential, and there is no corrective control for the situation. For the future, applying preventive controls is the only corrective control.

Table 1. The PreDeCo/CIA Table with general control types

	Confidentiality	Integrity	Availability
<b>Preventive</b>	Set access rights, encryption (encoding)	Encryption (hash, digital signature)	Multiplied resources
<b>Detective</b>	Monitoring the access rights, analysing access logs (accountability)	Integrity checker	Net statistics about the load balance, keep-alive messages about the system elements
<b>Corrective</b>	Nothing for the past, preventive controls	Install patches, update or upgrade the system, back	New resources, cutting malicious

- **Corrective integrity:** If the system is available, an exploit can hurt the integrity of the system. For example, a known bug can be exploited from a remote site and get unauthorized access to the system. If there is an available patch for this security bug, it must be installed. Otherwise, self-made controls must be applied or the service must be stopped.
- **Corrective availability:** In the worst case, the system would be down or would be under an extreme load. Installing or putting into operation new resources can help well-meaning requests. Cutting the lines from where the extreme load arrives must be applied against malicious requests.

## FUTURE TRENDS

Trends are mainly about standardization of widely used processes. There are many standards (ISO, 2004) and guidelines that are in use (RFC, 2004), but for nonsecurity experts there are three main ones: BS7799/ISO17799, COBIT, and common criteria. They can help a whole company or project team think, construct, develop, test and audit security of their company, product, or policies.

### BS7799 / ISO17799

Developed by the British Standards Institution (BSI) in conjunction with an international user group, BS7799 represents industry-developed standards on information security management. BS7799 has two parts: Code of Practice for Information Security Management (the standard code of practice that provides guidance on how to secure an information system), and Specification for Information Security Management Systems (which specifies the management framework, objectives, and control requirements for information security management systems).

The first part was replaced by the ISO17799 standard.

The second part of the standard remains a BS standard (BS 7799-2:2002) and has been aligned with other management systems standards, including the ISO 9000 and ISO 14000 series.

## COBIT

The COBIT (ISACA, 2003) has been developed as a generally applicable and accepted standard for good IT security and control practices that provides a reference framework for management, users, information system (IS) audits, and control and security practitioners.

COBIT, issued by the IT Governance Institute and now in its third edition, is increasingly internationally accepted as good practice for control over information, IT, and related risks. Its guidance enables an enterprise to implement effective governance over the IT that is pervasive and intrinsic throughout the enterprise. In particular, COBIT's management guidelines component contains a framework responding to management's need for control and measurability of IT by providing tools to assess and measure the enterprise's IT capability for the 34 COBIT IT processes. The tools include:

- performance measurement elements (outcome measures and performance drivers for all IT processes);
- a list of critical success factors that provides succinct, nontechnical best practices for each IT process; and
- maturity models to assist in benchmarking and decision making for capability improvements.

## Common Criteria

The Common Criteria for Information Technology Security (CC, 1998, 1999) is an international security evaluation certification that was designed to replace the various national evaluation schemes (e.g., FC/TCSEC, ITSEC, CTCPEC). The standard has three main parts and several hundreds pages. Implementing it in a project, for a product, or at a company requires security specialists

who involve in this work every member of the project, product, team, or company.

The standard with its three parts became an ISO standard under the number 15408. The titles are Introduction and General Model (part 1), Security Functional Requirements (part 2), and Security Assurance Requirements (part 3).

## CONCLUSION

This general model implements a method that can be applied by anyone who is sensitive to the security of his or her system. Just ask what kind of control and protection measure is included in the technology or application to manage the risk of attack against confidentiality, integrity, and availability. One can map the protections and take appropriate measures to manage the risks by filling out such a table by oneself. For those who want or must check their system or product against a standard before an audit, the two basic standard and the COBIT framework will help to reach their aims.

It is very important to remember that security through obscurity is dangerous, therefore it is better to plan, implement, and audit controls instead of applying obscure security methods.

## REFERENCES

- Anderson, R. (2001). Retrieved October 2004 from <http://www.cl.cam.ac.uk/~rja14/book.html>
- BS7799-2:2002, part two of British Standard 7799. Specification for Information Security Management). The Internet <<http://www.iso.ch>>
- CC. (1998). Common Criteria [IT standard]. Retrieved from <http://csrc.nist.gov/cc>
- CC. (1999). Common Criteria [IT standard]. Retrieved from <http://www.iso.org>
- Information Systems Audit and Control Association. (2003). COBIT: Control objectives for information and related technology [IT standard]. Retrieved from <http://www.isaca.org>
- International Organization for Standardization. (2004). Retrieved from <http://www.iso.org>
- NCR. (2002). IT security: Hackers, crackers, and thugs. Retrieved from [http://www.ncr.com/repository/articles/pdf/wcs\\_security.pdf](http://www.ncr.com/repository/articles/pdf/wcs_security.pdf)

RFC. (2004). Request for comments, de facto standards for the Internet, and many of them is security-related. Retrieved October 2004 from <http://www.ietf.org/rfc.html>

Schneier, B. (1996). *Applied cryptography*. New York: Wiley.

Schneier, B. (2000). *Secrets & lies (digital security in a networked world)*. New York: Wiley.

Schneier, B. (2003). *Beyond fear (Thinking sensibly about security in an uncertain world)*. New York: Copernicus Books.

Shore, D. (2004). The ITsecurity.com Dictionary+ of information security, Retrieved October 2004 from <http://www.ITsecurity.com/zones.htm?z=13>

Stoneburner, G., Goguen, A., & Feringa, A. (2002, July). Risk management guide for information technology systems. Retrieved October 2004 from <http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf>

## KEY TERMS (SHORE, 2004)

**Accountability:** One of the fundamental requirements of information security, accountability is the property that enables activities on a system to be traced to specific entities; who or which may then be held responsible for their actions. It requires an authentication system (to identify Users) and an audit trail (to log activities against Users). Accountability supports nonrepudiation, deterrence, fault isolation, intrusion detection and prevention, and after-action recovery and legal action (forensics).

**Audit:** (1) The process of compiling a list of all security relevant events. The list itself is called the “audit trail” and is essential and invaluable in ensuring accountability. (2) The process of compiling a list of all software and/or hardware installed on one or more PCs.

**Authentication:** A secure system requires that all users must identify themselves before they can perform any other system action. Authentication is the process of establishing the validity of the user attempting to gain access, and is thus a basic component of access control.

**Availability:** Requirement that information and/or services be available to an authorized user on demand. The maintenance of availability is one of the prime functions of a security system. Attacks against the availability of an information resource are called denial of service (DoS) attacks.

**Confidentiality:** Ensures that only those entities (both users and resources such as printers and other devices) that are authorized to access data may do so. It should not be confused with privacy, which is a different concept. Data whose confidentiality has failed is said to be compromised.

**Denial of Service (DoS):** An attack that is specifically designed to prevent the normal functioning of a system, and thereby to prevent lawful access to that system and its data by its authorized users. DoS can be caused by the destruction or modification of data, by bringing down the system, or by overloading the system's servers (flooding) to the extent that service to authorized users is delayed or prevented. Denial of Service attacks normally stem from external sources using telecommunications (such as via the Internet), or from disaffected or disgruntled employees who bear a grudge towards the company.

**Integrity:** More properly "data integrity," it is the property that the data in question has not been changed.

It is particularly important that integrity and confidentiality be combined, so that sensitive information can be neither altered without being read, nor read without being altered. Data whose integrity has failed is said to be corrupted.

**Policy:** The security policy is the collection of rules that define an organization's security objectives and how those objectives are to be achieved. The security policy must also be clearly and fully documented and enforced.

**Security Through Obscurity (Obfuscation):** Generally a derogatory term used when vendors seek to hide security details. The basic principle is that if no one knows any of the details of the security, then, equally, no one knows any weaknesses. The problem with this argument is that you can never be certain that crackers have not already found the weaknesses—better by far to be as open as possible so that good guys can find and help solve those problems.

# Semantic Enrichment of Geographical Databases

**Sami Faïz**

*National Institute of Applied Sciences and Technology, Tunisia*

**Khaoula Mahmoudi**

*High School of Communications-Tunis (SUPCOM), Tunisia*

## INTRODUCTION

The distributed Web-based multi-document summarization system is conceived to enrich semantically Geographic Databases (GDB) (Faïz, 1999; Scholl et al., 1996). In fact, in a traditional database, for instance, a city is described by its alphanumeric features: name, population count, and so forth; however, in a GDB, it is further described by spatial attributes which indicate its position (coordinates) in the space and its shape (point, line, polygon, etc.). Although the use of this myriad of information (alphanumeric and spatial data), the GDB suffers from the lack of an exhaustive set of information describing in a quasi-complete way the entities handled by it (Faïz, 2001). Hence, Geographic Information System (GIS) is not able to provide the end-user with information not fed into the GDB and that is not inherent to the application for which the GIS is designed (Bâzaoui, Faïz & Ben Ghezala, 2001, 2003; Faïz, Abbassi & Boursier, 1998). For instance, given a map displayed on the screen, it is not possible to get economic or historical information about cities for a given country whenever the GIS is concerned only with administrative boundaries. Having this idea in mind, our intention is to profit from the huge mass of information available online to enrich semantically a GDB. To fulfill this purpose and to manage the great amount of documents retrieved from the Web in a quick and convenient fashion, we adopted the Text Mining techniques (Tan 1999; Weiss, Apte & Damerau, 1999) and more precisely the summarization. Indeed, with the fast growth in the amount of textual information available online and the multitude of documents reporting almost the same thing, there is clearly a strong need for automatic summarization that copes with not only one document at one time but a set of topically similar ones.

Such systems are referred to as Multi-Document Summarization systems (MDS). While building such summarizers, there are many user requirements that have to be satisfied, in essence, the minimization of the redundancy and the coverage of all the information reported by the set of documents.

In fact, with the continuous growth in an astounding rate of information and the capacity of reading and analyzing that remains constant, the users are becoming amassed with a huge amount of information. Building up a distributed system is time consuming. To fulfill this objective, we used the Multi-Agent Systems (MAS). The distribution is also justified because MDS can be seen as a naturally-distributed problem owing to the fact that more than one entity is involved. Our intention is to speed up the summarization process while holding the main issues inherent to the problem.

## BACKGROUND

Some MDS systems will be outlined and the MAS paradigm will be presented.

### Multi-Document Summarization (MDS)

MDS (Lin & Hovy, 2002; Gees et al., 2000; Mani & Bloedoran, 1999; Regina, Kathleen & Michael, 2000) consists of condensing the content of a corpus of documents while coping with some issues, essentially, the coverage and the redundancy. The former concerns dealing with all the information conveyed by the whole collection. For the redundancy, one has to summarize the corpus of documents while not retrieving portions of text reporting the same information already included in the summary. In what follows, we outline some MDS approaches.

Gees et al. (2000) developed a summarizer that first creates individual summaries for all the documents in the set. Afterwards, these document summaries are grouped into clusters according to the similarity of their topics. Finally, for each cluster, a representative summary is selected. In the case of a summary based on a query submitted by the user, the representative summary is the one that has the most similarity with the concerned topic. In the case of a generic summary, the representative will be the one having words that occur frequently across all the summaries of the cluster.



In Radev, Jing, and Budzikowska (2000), the summarization process begins by clustering the articles into groups relative to similar events. Then, for each cluster, the relevance of the sentences is computed. In fact, the degree of relevance (from 0 to 10) for a given sentence to the general topic of the entire cluster is determined by picking out the most frequent words across the cluster. An utility of 0 means that the sentence is not relevant to the cluster, and a 10 marks an essential sentence. Then, each sentence ranked first according to this degree and beyond a fixed threshold is retained. Because many sentences have similar contents, one has to minimize the redundancy by filtering out the resulting sentences. This is based on the notion *cross-sentence informational subsumption*. It states that a sentence  $S_2$  subsumes  $S_1$ , if it has overlapping words with  $S_1$  and presents additional information. Hence, we have to include only one sentence of each class, according to the level of details desired. At the end of this process, each cluster is described by a set of relevant and non-redundant sentences.

Another method belonging to this field is the Maximal Marginal Relevance (MMR) Multi-Document Summarization (Goldstein et al., 2000). This method begins by segmenting the documents into passages. These may be sentences, sequences of sentences, or paragraphs. Then, the relevant passages to a given query are identified, using the cosine similarity metric. Thus, the passages under a fixed threshold are discarded. For the remainder of the passages (above the threshold), the MMR metric is applied. According to this measure, text passage has high marginal relevance if it is relevant to the query, while having minimal similarity to previously selected passages. The selected passages constitute the summary of the documents corpus.

## **Multi-Agent Systems (MAS)**

MAS (Barbuceanu, 1998; Briot & Demazeau, 2001; Mahmoudi & Ghédira, 2000) are generally regarded as being systems in which a group of autonomous agents interact to perform some set of tasks or satisfy some set of goals. According to Ferber, an agent is a hardware or software entity able to act on itself and on its environment. It has a partial representation of its environment and is able to communicate with other agents. It aims at an individual goal, and its behavior is the result of its observations, knowledge, abilities, and interactions it can have with other agents and with the environment (Ferber, 1997). These agents have several concerns to cope with, and they inhabit and interact within dynamic and not entirely predictable environments. Relevant examples of MAS include a group of agents moving a heavy object, telecommunication networks, and so forth.

These systems have proven their efficiency owing to the large number of publications which report their applications in heterogeneous domains.

## **MAIN THRUST OF THE ARTICLE**

The overall process and the main modules of the system will be detailed.

### **Overall Enrichment Process**

The semantic enrichment occurs when a user is looking for information about geographic entities. In fact, the GDB is queried at first to retrieve the data stored. Whenever the user is not satisfied by the response, the user starts up the summarization system. Thus, a mining of the online documents is triggered and will be over by returning the generated summaries.

The system relies on two kinds of agents: interface agent and summarizer ones. Every agent enjoys a simple structure independently of its type: acquaintances (the agents it knows), a local memory gathering its knowledge, a mailbox, storing the received messages that it will process later on.

The interface agent is responsible of launching the overall summarization process. It gets the set of documents resulting from an information retrieval task triggered by the user handling a map. Afterwards, it creates the summarizer agents whose number is equal to the one of the documents. These latter execute simultaneously a segmentation algorithm in order to determine the subtopics discussed along through the texts of their documents. Thus, each document is seen as nothing but a set of subtopics related to the main topic. Then, the interface agent carried a delegation task which aims to allocate to each subtopic a delegate (one of the summarizers) responsible for condensing the segments relative to a given subtopic. In fact, the summary of the corpus of documents is the merging of the partial summaries generated by the delegates. Thus, each delegate agent will extract the most salient pieces of texts retrieved from the set of segments under its jurisdiction. It considers one or all the segments and builds its, or their, Rhetorical Structure-Trees (RS-tree). The summary of a given subtopic under the responsibility of a delegate is generated by gathering the most relevant sentences.

### **Segmentation**

Each summarizer agent executes a segmentation algorithm in order to detect the subtopics discussed in its

document. To this end, we adopted the TextTiling algorithm (Hearst, 1997). The algorithm begins by converting the raw text to streams of tokens. These are grouped into sequences called token-sequence of size  $w$  (the number of tokens). These latter are in turn gathered into blocks of size  $k$  (number of the token-sequences). Similarity values are computed for every token-sequence gap number; that is, a score is assigned to token-sequence gap  $i$  corresponding to how similar the token-sequences from token-sequence  $i-k$  to  $i$  are to the token-sequences from  $i+1$  to  $i+k+1$ . Similarity values are computed for every token-sequence gap according to the following formula:

$$score(i) = \frac{\sum_t w_{t,b_1} w_{t,b_2}}{\sqrt{\sum_t w_{t,b_1}^2 \sum_t w_{t,b_2}^2}}$$

Where  $t$  is a token, and  $w_{t,b}$  is the weight (frequency) of  $t$  in block  $b$ .  $b_1$  and  $b_2$ , two text blocks, where,  $b_1 = \{token-sequence_{i-k}, \dots, token-sequence_i\}$  and  $b_2 = \{token-sequence_{i+1}, \dots, token-sequence_{i+k+1}\}$ .

Then, the depth at each gap between the blocks is computed to determine the deeper valleys that mark the segments relative to different subtopics. For each segment, the topic identification is fulfilled by considering the most frequent words as topics.

## Delegation

At this stage, the purpose is to affect a delegate agent to each virtual document. This latter is called so because it is not a result of an information retrieval task, but it is the output of gathering the segments dealing with the same subtopic, which are distributed among the summarizer agents.

In fact, the interface has to take into account the cost incurred by each delegation. Formally, the cost function is:

$$f(s) = \alpha workload + \delta communication$$

Where  $s$  is a given allocation;  $\alpha$  and  $\delta$  are the weight coefficients determined experimentally.

The workload is defined as follows for each delegate  $k$ :

$$workload = \sum_i \sum_j segment-size_{ij}$$

Where  $i$  ranges over the set of subtopics affected to agent  $k$ ,  $j$  ranges over the set of all agents tackling  $i$  and  $segment-size_{ij}$ , the size (number of sentences) of a seg-

ment retrieved from the text of the agent  $j$  and dealing with subtopic  $i$ .

For the communication, it is defined as:

$$communication = \sum_j \beta$$

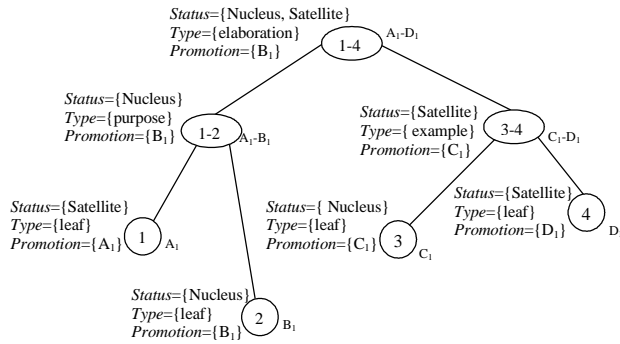
with  $\beta=1$  and  $j \in$  acquaintances of agent  $k$

It costs 0 if the agent  $k$  is a delegate of one of the subtopics it handled. Otherwise, it is equal to the total number of the agents dealing with the subtopic at hand. In fact, each subtopic stemming from the segmentation is considered as a group, and the agents that deal with this subtopic are its members. Hence, an agent can be simultaneously a member of more than one group whenever its document tackles more than one subtopic. The interface processes first the groups with the fewer members. For instance, it processes first the groups having cardinality equal to one. These groups have a unique candidate (the unique member) for the delegation. In fact, for each group (subtopic) the interface considers all its members ordered according to the increasing workload. We do so, not to overload some agents more than others, which is quite wasteful. Each summarizer agent already affected is removed from all the groups to which it belongs to give the opportunity to other agents to participate in the summarization process. In the case where a group has no member (all its members are removed that are already affected) and it is not yet allocated, the interface considers all the summarizers (its members as well as the rest of the summarizer agents) ordered according the increasing cost (value of  $f$ ). Thus, it affects the subtopic at hand to the agent having the minimum cost. Hence, a delegate can be responsible for condensing a subtopic that it does not tackle initially (according to its native document). Finally, any summarizer agent can be a delegate of 0, 1, or several subtopics that it has to condensate.

## Text Extraction

Once the delegation is over, each delegate uses one of the following alternatives for each subtopic it governs: arbitrarily selects any of the segments, selects the longest segment because we hope that it gives more details of the subtopic, and considers all the segments. Hence, from the selected segments, each delegate derives the most important parts that best summarize the subtopics by building the RS-Trees (Marcu, 1999). In fact, an RS-Tree is a binary tree whose leaves denote elementary textual units and whose internal nodes correspond to contiguous text spans. Each node has a status (nucleus or satellite), a type (the rhetorical

Figure 1. RS-Tree



relation that holds between the text spans), and a salience or promotion set (the set of units which are the most important part of the text that is spanned by that node). A relation is maintained between two units, one of which is more important to the author point of view than the other. The former is referred to as the nucleus and the latter the satellite. By removing the satellite, the text still remains comprehensible. Then, generating a summary can be accomplished by merely ordering important units in the original text by considering that the units that are promoted closer to the root are more important than those that are promoted less close. To illustrate this approach, we give the following piece of text dealing with reform of irrigation policy and water conservation and its simplified RS-Tree as shown in Figure 1.

*[To conserve water resources and encourage demand management in the irrigation sector,<sup>A1</sup>] [a national water saving strategy was implemented.<sup>B1</sup>] [As part of the strategy, a number of reforms were introduced in the past few years,<sup>C1</sup>] [for instance, the promotion of water users' associations, the increase in the price of irrigation water, etc.<sup>D1</sup>]*

By retrieving the most important parts, we obtain this ordering: B<sub>1</sub> > C<sub>1</sub> > A<sub>1</sub>, D<sub>1</sub>. B<sub>1</sub> is retrieved first because it is the root of the tree, followed by C<sub>1</sub>, which is found one level below B<sub>1</sub>, and so on. By applying this criterion to each RS-Tree associated with each selected segment, each delegate retrieves the most relevant portions of the segments for each subtopic under its jurisdiction.

### Agents Communication

Once a summarizer achieves the segmentation, it sends to the interface agent the list of its subtopics discussed

through its segments via **m<sub>1</sub>: ListSubTopics (Summarizer, Int, SubTopic-set)**.

The syntax used in this message and the other ones is **MessageLabel (idS, idR, message-parameter)**. *MessageLabel* is the type of message sent by *idS* to *idR*. In fact, the eventual values of *idS* or *idR* are *Int*, *Delegate*, and *Summarizer*. *Int*, for the Interface, *Delegate*, any delegate, and finally, the *Summarizer* is any summarizer agent. *Message-parameter* means what is sent by the sender to the receiver.

Then, the interface forms the groups which are the subtopics tackled in all the corpus and affects to each of its members the agents dealing with the corresponding subtopic. Finally, the interface makes the delegation decision. Whenever the delegation is over, the interface agent sends a notification message, **m<sub>2</sub>: Delegation (Int, Delegate, SubTopic-set)**, to inform the concerned agents about the delegation decision.

There are two possible cases: either the agent is a delegate of one of the groups to which it belongs or it is a delegate of a group to which it does not belong (it could be a delegate of groups to which it belongs as well as the groups to which it does not belong).

For the first case, the delegate agent sends a message **m<sub>3</sub>: SubTopicRequest (Delegate, Summarizer, SubTopic-set)** asking its acquaintances (according to the subtopics) to send the pieces of texts dealing with the specified subtopics. The receivers in turn send the message **m<sub>4</sub>: TextSubTopic (Summarizer, Delegate, Text-set)** as a reply to the **SubTopicRequest** query.

In the second case, the communication flow is as follows: The delegate agent sends a request message to identify the members of the groups under its responsibility, **m<sub>5</sub>: ListMembers (Delegate, Int, SubTopic-set)**. This message is sent on behalf of the delegate agent to the interface, which knows all the members of the society. The interface agent responds by providing the list of the concerned agents, **m<sub>6</sub>: SubTopicMembers (Int, Delegate, Member-set)**. Then and after identifying all the members concerned with the specified subtopics, the delegate sends **m<sub>3</sub>** to collect all the texts (segments) and waits until it receives **m<sub>4</sub>** messages.

Whatever the situation, a delegate starts the summarization of its virtual document if it receives all the replies to the **m<sub>3</sub>** messages for a given subtopic; otherwise, it waits. If a delegate achieves all the work affected to it, it sends **m<sub>7</sub>: Summary (Delegate, Int, Summary-set)** to provide the interface with the summary of each subtopic under its responsibility.

The whole process is over if the interface is satisfied, a situation reached whenever all the summarizer agents are satisfied, too. A summarizer agent is said to be satisfied if it achieves all the work affected to it, or else it is said unsatisfied and continues the summarization.

## FUTURE TRENDS

Semantic enrichment of geographical databases is concerned with supplying additional information about geographic entities without taking into account the spatial aspect (shape and topological relations); only the alphanumeric attributes are handled. One perspective is to go beyond this limitation and to profit from the spatial aspect by considering the neighborhood of the entities. For instance, whenever the results of the current mining are not satisfactory enough, one can extend the search space to the adjacent entities. In fact, mining information about the neighbors may give knowledge more valuable than mining the documents related to the entity itself.

## CONCLUSION

Our system aims to enrich geographical databases. GIS provides information retrieved from the GDB; thus, any data not stored within is not available. To make these databases rich sources of information, allowing any user to know almost everything about the displayed entities, we mined data available on the Web. This mining is achieved via the summarization technique and, more precisely, the MDS by collaborating a set of agents. There are essentially two classes of agents, namely, interface agent and summarizer ones. These agents interact to lead the system to the optimal summary.

## REFERENCES

Bâzaoui, H., Faïz, S., & Ben Ghezala, H. (2001). Spatial data warehouses and exploration techniques. *Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications*, Beirut, Liban.

Bâzaoui, H., Faïz, S., & Ben Ghezala, H. (2003). CASME: A case tool for spatial data marts design and generation. *Proceedings of the 5th International Workshop on Design and Management of Data Warehouses (DMDW'03) in conjunction with the 29th International Conference on Very Large Databases (VLDB'03)*, Berlin, Germany.

Barbuceanu, M. (1998). Negotiation as interactive exchange of constraint about agent behavior. *Proceedings of the International Workshop on Multi-Agent Systems*, Boston.

Barzilay, R., Mckeown, K., & Elhadad, M. (2000). Information fusion in the context of multi-document summariza-

tion. *Proceedings of the 37th Annual Meeting of the ACL*, (pp. 550-557), MD.

Briot, J.P., & Demazeau, Y. (2001). *Principals and architecture of multi-agent systems*. Paris: Edition Hermes Science.

Faïz, S. (1999). *Geographic information system: Quality information and data mining*. Editions C.L.E.

Faïz, S. (2001). Spatial data mining, spatial data warehousing and spatial OLAP: application to geographic data quality. *Proceedings of the 4th Tunisian Interdisciplinary Workshop on Science & Society (TIWSS)*, Tokyo, Japan (pp. 18-22).

Faïz, S., Abbassi, K., & Boursier, P. (1998). Applying datamining techniques to generate quality information within geographical databases. In Jeansoulin & Goodchild (Eds.), *Data quality in GI*. Paris: Editions Hermes.

Ferber, J. (1997). *Les systèmes multi-agents vers une intelligence collective*. Edition InterEditions.

Gees, C.S., Strzalkowski, T., Wise, G.B., & Bagga, A. (2000). *Evaluating summaries for multiple documents in an interactive environment*. General Electric, Corporate R&D.

Goldstein, J., Vibhu, M., Carbonell, J., & Kantrowitz, M. (2000). Multi-document summarization by sentence extraction. *Proceedings of the ANLP/NAACL Workshop on Automatic Summarization*, Seattle, Washington.

Hearst, M.A. (1997). TextTiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1), 33-46.

Lin, C.Y., & Hovy, E. (2002). From single to multi-document summarization: A prototype system and its evaluation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, (pp. 457-464), Philadelphia.

Mahmoudi, K., & Ghédira, K. (2000). Distributed rescheduling for the workforce management dynamic aspect. *Proceedings of the 3rd Ibero American Workshop on Distributed Artificial Intelligence and Multi-Agent Systems*, Atibaia, Sao Paulo, Brazil.

Mani, I., & Bloedoran, E. (1997). Summarizing similarities and differences among related documents. *Proceedings of the RIAO-97*, Montreal, Canada (pp. 373-387).

Marcu, D. (1997). *Discourse trees are good indicators of importance in text*. In Mani & Maybury (Eds.), *Advances in automatic text summarization* (pp. 123-136). MIT Press.



Radev, D.R., Jing, H., & Budzikowska, M. (2000). Centroid-based summarization of multiple documents: Sentence extraction. *Proceedings of the ANLP/NAACL Workshop*, 21-19.

Scholl, M., Voisard, A., Pelous, J.P., Raynal, L., & Rigaux, P. (1996). *SGBD géographiques*. International Thomson Publishing France.

Tan, A. (1999). Text mining: The state of the art and the challenges. *Proceedings of the PAKDD'99 Workshop on Knowledge Discovery from Advanced Databases*, Beijing, Singapore.

Weiss, S., Apte, C., & Damerau, F. (1999). Maximizing text-mining performance. *IEEE Intelligent Systems*.

## **KEY TERMS**

**GDB:** Geographic database (GDB) is a database integrated into the GIS, storing spatial and alphanumeric data.

**GIS:** Geographic Information System (GIS) is a computer system capable of capturing, storing, analyzing, and displaying geographically referenced information.

**MAS:** Multi-Agent Systems (MAS). In a distributed universe, the systems that shelter agents, are called Multi-Agent Systems. An agent, is a hardware or software entity able to act on itself and on its environment.

**MDS:** Multi-Document Summarization (MDS) is the process of distilling the most important information from a corpus of documents.

**RS-Tree:** A binary tree, which describes the rhetorical structure of every coherent discourse.

**Text Mining:** Known as text data mining or knowledge discovery from textual databases, and refers to the process of extracting interesting and non-trivial patterns or knowledge from text documents.

**TextTiling:** A technique for automatically subdividing texts into multi-paragraph units that represent passages, or subtopics.



# Semantic Information Management

**David G. Schwartz**

*Bar-Ilan University, Israel*

**Zvi Schreiber**

*Unicorn Solutions Inc., USA*

## INTRODUCTION: THE ENTERPRISE DATA PROBLEM

The need to manage enterprise data has been coming into increasingly sharp focus for some time. Years ago, data sat in silos attached to specific applications. Then came the network, with data becoming available across applications, departments, subsidiaries, and enterprises. Throughout these developments, one underlying problem has remained unsolved: Data resides in thousands of incompatible formats and cannot be systematically managed, integrated, unified, or cleansed.

To make matters worse, this incompatibility is not limited to the use of different data technologies (e.g., flat file, COBOL, IMS, Relational, XML, etc.) or to the multiple different “flavors” of each technology, such as the different relational databases (Oracle, DB2, SQL Server, Sybase, etc.) in existence. The most challenging incompatibility arises from *semantic differences*. Each data asset is set up with its own worldview and vocabulary—known as its schema. This incompatibility exists even if both assets use the same technology.

For example, one database has a table called “client,” intending this to include channel partners, and subdivides customers into individuals and institutions; the other data asset refers to the same concept as a “patron” (although not including channel partners) and subdivides “patrons” into individuals, corporations, government, and nonprofit groups. To make matters worse, “patron” excludes international clients, despite the fact that this is not explicitly mentioned in any documentation and the original developer retired five years ago.

In a larger enterprise, this problem may be multiplied by thousands of data structures located in hundreds of incompatible databases and message formats. And the problem is growing; enterprises continue to acquire subsidiaries, reengineer processes, and integrate with partners. Moreover, developers are continuing to write new applications and to create new databases based on requests from business users without worrying about overall data management issues.

Therefore, it is imperative to find an efficient way to manage multiple applications and data sources. This

requires a shift from an application-centric view of IT to a data- and information-centric view. And it requires a focus on the business meaning—or semantics—of the data. Focusing on a singular business meaning results in the effective use of a single language throughout the enterprise—the common business language approach.

## BACKGROUND

### Impact of the Data Problem

The enterprise data problem has a strong measurable impact on a company’s bottom line (Schreiber, 2003a).

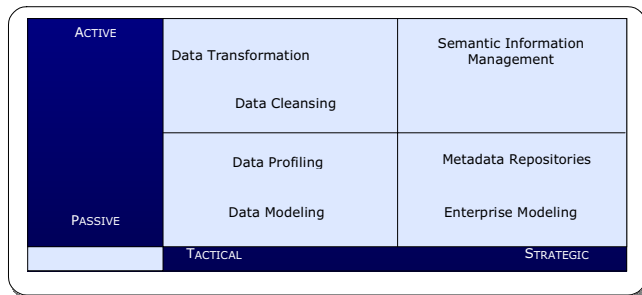
First, the fragmented data environment inevitably leads to business *information quality problems*, causing businesses to mishandle information, customer relationships, and internal operations.

Second, the data problem creates a situation which all but prevents the agility that is critical to a modern enterprise responding to a constantly changing environment. Whether application deployment, business process reengineering, or mergers and acquisitions are involved, IT is unable to respond to a dynamic environment due to the fragmented and delicate nature of data assets and hard-coded scripts keeping assets communicating with one another. This is a significant barrier to moving towards the real-time—or zero-latency—enterprise (Khosla & Pal, 2002).

Finally, IT remains unnecessarily inefficient so long as it lacks a strategic approach to data management. In the meantime, IT deals with the frustrating and costly challenge of administering databases (some of which are redundant), mapping each database multiple times and writing and manually maintaining point-to-point translation scripts.

Data needs to be systematically managed if it is to have long-term value to the enterprise. Specifically, data management must start by elevating *data* into *information* by explicitly capturing the meaning and context of the data. This process is known as *data semantics* (Sheth, 1995).

Figure 1. Positioning of data management disciplines



### An Overview of Existing Approaches

A partial solution is a *metadata repository* containing information about each data asset. Repositories act as catalogs and include technical information about the assets, their structure, how they are used, and who is responsible for them (DeMarco, 2000). But a passive catalog does not provide a formal understanding of data or automated support for translating and cleansing the data.

A tactical solution to data quality (Strong, Lee, & Wang, 1997a, 1997b; Wand & Wang, 1996) involves *data profiling* and *data cleansing*. While these are important tools within the overall approach to data management, enterprise information quality will never be achieved one database at a time. The real problem lies in ensuring that there is an agreed-upon understanding of each data asset and of its relationship to other data assets. Data assets must be designed, created, maintained, integrated, and decommissioned with attention to the wider quality aims. This is accomplished by understanding data as part of an overall information architecture (Schreiber, 2003b) and by understanding and validating the data with respect to a single agreed-upon business worldview and a series of business rules.

*Data modeling* typically supports better database design. But data models are usually entity-relationship diagrams (Chen, 2002) with limited business depth, lacking generalization/specialization capabilities (also known as inheritance, subtyping, or polymorphism), and not capturing business rules. They are also technically suitable only for modeling relational databases one at a time, while IT increasingly uses XML, especially for messaging. Furthermore, data models are typically tightly coupled to a specific relational database and are not generally used to promote a common understanding of multiple data assets.

### SEMANTICS: TOWARDS A COMMON BUSINESS LANGUAGE

Companies will always struggle with a large number of physically different data formats. While a common data format will likely never be achieved, the key to efficiently managing data is to establish a *common understanding*. This is the idea of *semantics*, bridging nomenclature and terminological inconsistencies to comprehend underlying business meaning in a unified manner. Semantics can be achieved by formally capturing the meaning of data. This is accomplished by relating physical data schemas to concepts in an agreed-upon model of the business. Wand and Wang (1996) present some of the foundations for building improved data quality on an ontological basis.

This central model of the business is called an *information model* (Sheth & Kalinichenko, 1992). The information model does not reflect any specific data model but rather reflects the agreed-upon business view, business vocabulary, and business rules which will provide a common basis for understanding data.

Semantics builds upon traditional informal metadata (Bernstein & Bergstaesser, 1999) and captures the formal meaning of data in agreed-upon business terms. For example, the information model might capture the official business concepts of a “customer” and the more specific concepts of a “business customer” and an “individual customer.” A semantic mapping will then relate physical data schemas to this information model. For instance, a semantic mapping might capture the fact that an information model has a universally accepted concept called “individual customer” that is called “client” by a relational database table, “customer” by an XML schema, and “CUST3” by a COBOL copybook. Thus the semantic mapping formally captures the meaning of the data by reference to the agreed-upon business terminology.

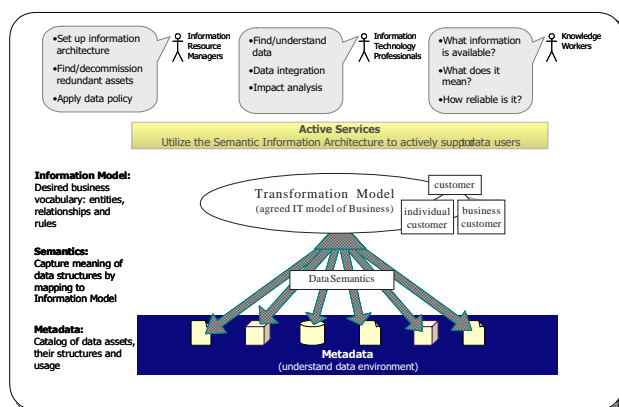
Why take the trouble to capture data semantics? Two reasons. Tactically, semantics saves time by capturing the meaning of data once. Without semantics, each data asset will be interpreted multiple times by different developers as it is designed, implemented, integrated, cleansed, extended, extracted (for a warehouse), and eventually decommissioned. This independent interpretation will be time-consuming and error-prone. With semantics, the data asset is mapped and interpreted only once. Second, new assets can be generated from the information model so that they use official business terminology from the outset and never require mapping.

The most significant impact of semantics is a strategic one. Semantics turns hundreds of data sources into one coherent body of information. The semantic architecture includes a record of where data is and what it means. Using this record of which business information is represented in each data asset, it becomes possible to automate the search for overlap and redundancy. The information model provides the basis for creating new data assets in a consistent way and serves as a reliable reference for understanding the interrelationship between disparate sources and for automatically planning how to translate between them. Finally, semantics provides a central basis for impact analysis and for the smooth computer-aided realization of change.

Semantics is not only being applied to managing enterprise data. The World Wide Web Consortium is working to reinvent the Web as the Semantic Web (Berners-Lee, Hendler, & Lasilla, 2001; Schwartz, 2003), where information will carry computer-readable meaning.

While some informal data semantics has existed for decades in data dictionaries in home-grown solutions and ETL tools, a more formal semantics is key to a modern information architecture enabling an effective strategic approach to data management.

Figure 2. Semantic information management (SIM)



## SEMANTIC INFORMATION MANAGEMENT: CORE PRINCIPLES



Core elements of semantic information management (SIM) may be summarized as follows (see also Figure 2).

- Metadata—*Know your data.*
- Information Model—*Know your business.*
- Data Semantics—*Understand your data.*

### Metadata

Before data assets can be understood, they must be cataloged. Metadata should include the asset’s schema as well as information about an asset’s location, usage, origin, relationship to other assets, rules associated with it, and assignment of ownership. Some of this metadata may be scanned automatically from assets such as relational databases or from existing sources of metadata.

### Information Model

The information model is a rich central model of the business or of a domain within the business. A traditional data model or object model may serve as the basis for an information model, but ideally data models should be extended to a full *ontology*. An ontology literally means *a study of what exists* or as Gruber (1993) succinctly put it: “the specification of a conceptualization.” An ontological information model is therefore typically richer than a data model in its view of the business, including different levels of generalization/specialization and a layer of business rules in addition to the traditional entities and relationships. This richness allows the information model to serve as an authoritative reference by which meaning is given to multiple data assets, regardless of format or technology.

A well-organized information model will model business in five layers:

- **Organizational Layer:** Divides and subdivides the information model into different packages reflecting different parts of the business, such as customer, product, etc. reflecting ownership of different parts of the model.
- **Entity Layer:** Captures the entities, *things*, or *classes* that play a role in each area of the business. Examples include people, documents, or products. The entity layer should capture the business at different levels of detail or specialization. The general concept of “product,” a specific cat-

egory of products, or a specific product might all be captured as entities.

- **Property/Attribute Layer:** Describes the vocabulary used to describe each entity and to relate entities. For example, it captures the fact that every customer is associated with a contact person or that a product is associated with a price. Properties are also known as *relationships*.
- **Business Rule Layer:** Allows the information model to centrally capture business rules relating alternative vocabularies. Without this, these business rules would be hard-coded in applications and scripts around IT and would be impossible to track and change. Examples of rules are lookup tables for alternative product codes or logical/arithmetic rules relating alternative vocabularies. This layer might relate the concept of annual revenue to quarterly revenue or capture the logic for the different ways in which the marketing and service departments segment customers.
- **Descriptor Layer:** Ensures that all concepts in the other four layers are documented in a structured way, providing definition, synonyms, alternatives, examples, and, where needed, foreign language names.

Note that layers 1-3 are entirely compatible with UML class diagrams, popular with application developers. However, the business rule layer is a key differentiator. Whereas UML emphasizes application dynamics, an ontological information model has a different purpose—capturing business logic.

In summary, an ontological information model leverages existing data and application modeling skills to allow a richer and more intuitive view of business logic.

## DATA SEMANTICS

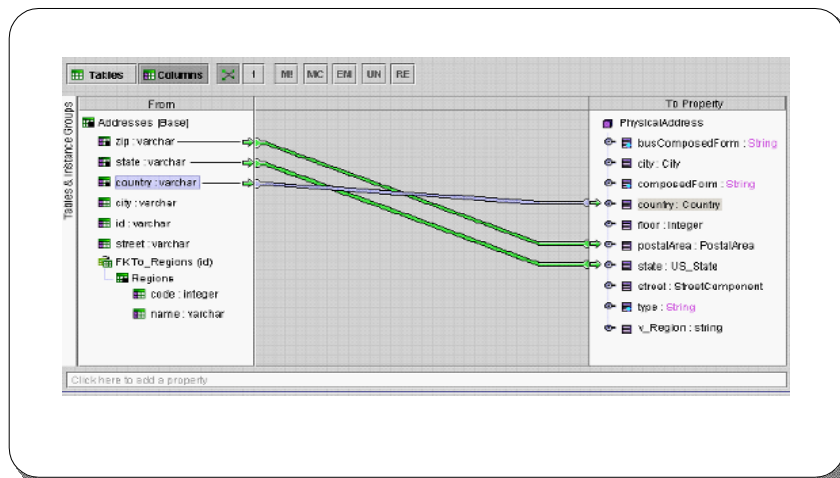
Semantics captures the formal meaning of data. It is achieved by *mapping* (or *rationalizing*) the data's schema to the information model.

Any database or message format with a schema can be mapped, including relational databases, XML, older hierarchical databases, network databases, and COBOL copybooks. Data that is structured without a schema (e.g., EDI messages or flat files) can be parsed and then mapped.

Software can aid the mapping process (see Figure 3) using type information, foreign keys, and even name similarities to suggest matches and to provide an efficient graphical environment. However, mapping will never be totally automatic; only a database administrator or other expert will know how to interpret data accurately.

Semantic mapping creates immediate savings. Having mapped an asset once to an information model, its relationship to all other assets may be inferred automatically. *Every asset is therefore mapped only once*, in contrast to the current situation in which every data asset is mapped many times, often using inappropriate tools such as MS Word or Excel.

Figure 3: Computer-aided semantic mapping of a relational database to an information model





## FUTURE TRENDS

Utilizing semantic information management can have a direct impact on data management, integration, and quality. Having a common understanding allows an enterprise to achieve its key strategic goals of managing data assets, integrating data, and improving data quality. In addition, these goals are achieved while addressing the entire body of information using standard business terms rather than grappling with hundreds of specific data formats.

### Data Management

- **Data Standards:** Semantic information management allows new databases and message formats to be directly generated from the information model so that they reflect an agreed-upon business terminology. This is particularly useful as enterprises move to generate internal XML schema messaging standards.
- **Data Discovery:** Data semantics provides a single business lens onto data assets. Managers may use the semantic mappings to discover the data assets covering a particular data concept and to direct developers to the correct assets.
- **Eliminating Redundancy:** Data discovery exposes redundancy between data assets. Redundancy not only causes unnecessary development and operational expense but also creates potential for data inconsistency and is therefore the root of many serious enterprise data quality problems.
- **Security:** Data discovery enables uniform security and privacy policies. Once a particular business concept is highlighted as sensitive, data discovery highlights all the data assets which require securing.
- **Influencing Developers:** Publishing metadata enables developers to find and reuse existing assets rather than invest time creating new ones. It also ensures that they are aware of relevant data usage restrictions and policies. Published semantics goes further in enabling engineers to accurately understand data, thus increasing their productivity and code quality. Finally, semantic information management code generation is attractive to developers, which in turn encourages them to store business rules and transformation code in a central way, thereby making maintenance more effective.
- **Increasing Data Services:** Semantic information management comes with active capabilities that increase the central data services or data

architecture group's value to the rest of the enterprise.

### Data Integration

- **EAI:** Enterprise application integration products (e.g., IBM WebSphere MQ, Tibco, WebMethods, Vitria, and SeeBeyond) provide an important platform for integrating applications and especially for facilitating messaging and workflow. However, these can be deployed and maintained far more cheaply and flexibly if their translation scripts are auto-generated rather than manually coded.
- **ETL/BI:** On the informational side of IT, mature tools exist to load warehouses and to perform business intelligence analysis of the warehouse. But when designing a warehouse, SIM provides the most reliable methodology for finding and interpreting sources, cleansing them according to central rules, planning transformation scripts, and creating a warehouse schema that reflects business reality.
- **Corporate Portals:** Nowhere is quality information and correct business vocabulary more important than in the corporate portal. Many good portal products provide the portal runtime, but SIM identifies the data sources and generates the data translations, ensuring accurate data presentation.
- **Version Changes:** Changing a database schema may have business importance but can be costly and risky. With a SIM approach, the old and new schemas can both be mapped to the information model. This allows the data migration scripts to be inferred automatically and enables all queries acting on the database to be identified and updated automatically to reflect the new schema.

### Data Quality

- **Unambiguous Meaning:** SIM contributes to data quality by ensuring that data has a formal and unambiguous business-oriented meaning. This helps to ensure that data is not misinterpreted either by its ultimate business audience or by developers writing queries to the data.
- **More Accurate Data Translations:** SIM can be used to automate the creation of accurate data translation scripts. These scripts are the critical links ensuring the integrity of data as it travels



through the enterprise. Most importantly, the architecture can be used to update these transformations in a consistent way when a change occurs in a data source or a business rule.

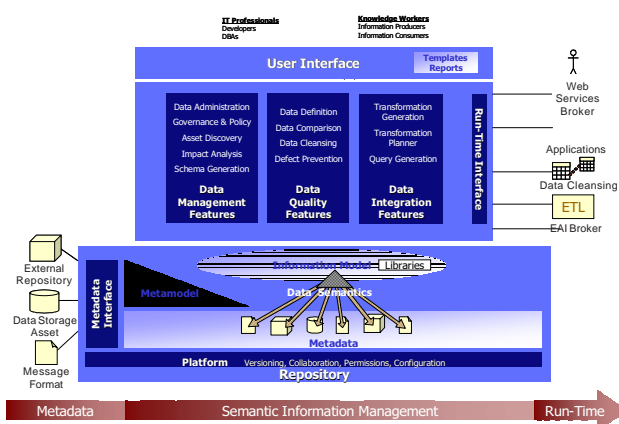
- **Data Consistency:** A common understanding of data allows incompatible data formats to be compared automatically, revealing hidden inconsistencies and overlap.
- **Central Approach to Data Rules and Cleansing:** Enterprise information quality cannot be achieved by cleansing one database at a time. Business rules and constraints need only be captured once in the information model for all data sources—regardless of their format—to be automatically validated against the same central set of rules.

## A SYSTEM FOR SEMANTIC INFORMATION MANAGEMENT

Semantic information management will be supported by an appropriate suite of products which should be fully integrated. Key components (as shown in Figure 4) should include:

- **Metadata:** A repository for storing metadata on data assets, schemas, and models associated with the assets.
- **Data Semantics:** Integrated tools for ontology modeling in order to support the creation of an information model (capabilities should include compatibility with entity-relationship and UML

Figure 4. Logical architecture diagram for semantic information management



- **Data Management Services:** The system should use the information model standard business terminology as a lens through which data is managed. Data management should include the ability to author and edit the information model, discover data assets for any given business concept, administer data, create reports and statistics about data assets, test and simulate the information model, and analyze impact in support of change.
- **Data Integration Services:** The system should automatically generate code for queries and data transformation scripts between any two mapped data schemas, utilizing the common understanding provided by data semantics.
- **Data Quality Services:** In order to provide a systematic approach to data quality, the system should support the identification and decommissioning of redundant data assets. It should support comparison for ensuring consistency among semantically different data and validation/cleansing of individual sources against the central repository of rules.
- **Metadata Interface:** The system must be able to collect metadata and data models directly from relational databases and other asset types and to exchange metadata with other metadata repositories. Similarly, the metadata and models accumulated by the system must be open to exchange with other systems through the use of adaptors and standards such as XMI (XML Metadata Interchange standard).
- **Runtime Interface:** A key differentiator of the semantic information technology is the active data integration capabilities. The runtime interface ensures that queries, translation scripts, schemas, and cleansing scripts generated automatically by the system may be exported using standard languages (such as SQL and XSLT) to runtime environments, including off-the-shelf ETL, EAI, and data cleansing products.
- **User Interface:** The user interface should include a rich thick-client for power users in the data management group. A customizable Web interface is usually best for developers and business users who are primarily concerned with reading and using the metadata but not creating it.
- **Platform:** The system should include a platform supporting version control, collaboration, per-

mission management, and configuration for all metadata and active content in the system.

## CONCLUSION

Semantics inspires a vision in which data carries unambiguous business meaning that can be found, aggregated, and used accurately and flexibly without prior knowledge of the data's specific format. Semantic information management delivers the benefits of semantics and a common business language to enterprise IT.

Semantic differences between these data formats create an environment with poor business information, lack of business flexibility, and high IT costs.

Existing solutions are insufficient. Metadata repositories are important but do not create sufficient value without semantics. Data modeling is usually limited to relational databases treated one database at a time. Data cleansing is an important way to improve a database but does not address the strategic problem of obtaining quality across hundreds of data assets.

Semantic information management addresses the core of the problem by capturing the precise meaning of data in agreed-upon terms. Key elements of the architecture are metadata (knowing your data), an information model (knowing your business), and data semantics (understanding your data).

Semantic information management creates value by delivering higher quality business information, providing the flexibility to support business change, and making IT costs lower and more predictable. It can be introduced gradually to specific projects and eventually extended to the entire enterprise.

With semantic information management, the enterprise can strive for an environment in which everyone speaks the same business language, data carries unambiguous business meaning, and the data environment is managed and integrated at will. This is accomplished while providing high quality information to the business and allowing the enterprise to adapt itself in real time.

## REFERENCES

- Berners-Lee, T., Hendler, J., & Lassila, O. (2001, May). The Semantic Web. *Scientific American*.
- Bernstein, P. A., & Bergstaesser, T. (1999). Meta-data support for data transformations using Microsoft Repository. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 26(4), 9-14.
- Chen, P. P. (2002, June). Entity-relationship modeling: Historical events, future trends, and lessons learned. In M. Broy & E. Denert (Eds.), *Lecture notes in computer science: Software pioneers and their contributions to software engineering* (pp. 100-114). Berlin, Germany: Springer-Verlag.
- DeMarco, D. (2000). *Building and managing the meta data repository: A full lifecycle guide*. New York: Wiley.
- Friedman, T. (2004, April). The importance of a common business language. *Unicorn*. Retrieved from [http://mediaproducts.gartner.com/webletter/unicorn\\_issue1/index.html](http://mediaproducts.gartner.com/webletter/unicorn_issue1/index.html)
- Gruber, T. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), 199-220.
- Khosla, V., & Pal, M. (2002). *Real time enterprises: A continuous migration approach*. Retrieved from <http://www.kpcb.com/files/bios/RTEWHITEPAPER.pdf>
- Schreiber, Z. (2003a). Applying the Semantic Web vision to enterprise data management: A case study. *12th International World Wide Web Conference*.
- Schreiber, Z. (2003b, October). Semantic information architecture: Creating value by understanding data. *Data Management Review*.
- Schwartz, D. G. (2003, May/June). From open IS semantics to the Semantic Web: The road ahead. *IEEE Intelligent Systems*, (pp. 52-58).
- Sheth, A. (1995). Data semantics: What, where and how?. In R. Meersman & L. Mark (Eds.), *Proceedings of the Sixth IFIP Working Conference on Data Semantics (DS-6)*. London: Chapman & Hall.
- Sheth, A., & Kalinichenko, L. (1992). Information modeling in multidatabase systems: Beyond data modeling. *Proceedings of the First International Conference on Information and Knowledge Management*.
- Strong, D. M., Lee, Y. W., & Wang, R. Y. (1997a). Data quality in context. *Communications of the ACM*, 40(5), 103-110.
- Strong, D. M., Lee, Y. W., & Wang, R. Y. (1997b, August). 10 potholes in the road to information quality. *IEEE Computer*, (pp. 38-46).
- Wand, Y., & Wang, R. Y. (1996). Anchoring data quality dimensions in ontological foundations. *Communications of the ACM*, 39(11), 86-95.

## KEY TERMS

**Data Semantics:** A reflection of the real world that captures the relationship between data in a database, its use in applications, and its corresponding objects in the real world. Data semantics requires some form of agreement between the different agents (human and computer) that interact with and use the data. The field of data semantics deals with understanding and developing methodologies to find and determine data semantics, represent those semantics, and enable ways to use the representations of the semantics.

**Data Quality:** Ensuring that data supplied is fit for use by its consumer. Elements of data quality can extend to include the quality of the context in which that data is produced, the quality of the information architecture in which that data resides, the factual accuracy of the data item stored, and the level of completeness and lack of ambiguity.

**Information Model:** A rich central model of the business or of a domain within the business. A traditional data model or object model may serve as the basis for an information model, but ideally data models should be extended to a full ontology.

**Metadata:** Metadata includes a data asset's schema as well as information about an asset's location, usage, origin, relationship to other data assets, rules associated with it, and assignment of ownership.

**Metadata Repository:** An information store containing information about each data asset. Repositories act as catalogs and include technical information about data assets, their structures, how they are used, and who is responsible for them.

**Ontology:** A taxonomic representation of terms and their agreed-upon associated meanings. A full ontology will include the formally defined relationships between terms in that ontology. An ontology should be all-encompassing, embracing every term within a universe of discourse.

**Semantic Information Architecture:** An architecture consisting of three core parts: metadata, an information model, and data semantics. The metadata facilitates a basic understanding of the organization's data; the information model provides a conceptualization and representation of the business, based on ontology, to provide a formal specification of the real world; and the data semantics map the data sources to the information model to capture meaning, as result of which we can understand the data.

# Semantically Modeled Enterprise Databases

**Cheryl L. Dunn**

*Grand Valley State University, USA*

**Severin V. Grabski**

*Michigan State University, USA*

## INTRODUCTION

A semantically modeled enterprise database is a reflection of the reality of the activities in which an enterprise engages and the resources and people involved in those activities. Many organizations have invested immense sums of money in enterprise resource planning systems (ERP) and associated “bolt-on” applications such as customer relationship management (CRM) and advanced planning systems (APS). A significant portion of the value of these systems is in the integrated database and associated data warehouse. To maximize value, the database should serve as a semantic representation of the organization. Otherwise, relevant information needed to reflect the organization’s activities may be omitted or may be stored in such a way that the underlying reality is hidden or disguised and is therefore of no use to decision makers.

Semantically modeled enterprise databases require their component objects to correspond closely to real-world phenomena and preclude the use of artifacts as system primitives (Dunn & McCarthy, 1997). Semantically modeled enterprise information systems allow for full integration of all system components centered on a single integrated database and facilitate joint use of information by decision makers. Researchers have advocated semantically designed information systems because they provide benefits to individual decision makers (Dunn & Grabski, 1998, 2000) and because they facilitate organizational productivity and interorganizational communication (Cherrington, Denna, & Andros, 1996; David, 1995; Geerts & McCarthy, 2001a).

Ontologically based systems with common semantics are necessary to facilitate interorganizational information systems (Dunn, Cherrington, & Hollander, 2005; Geerts & McCarthy, 2001b, 2003). Such systems are increasingly necessary as business-to-business e-commerce becomes a major component of the economy. Presently, most interorganizational data is sent via electronic data interchange (EDI), which requires very strict specifications as to how the data are sequenced and requires some investment by adopting organizations. Knowledge inherent in these systems is limited at best. Trading partners who

implement systems based on the same underlying semantic model may eliminate many of the current problems.

ERP systems have been defined as “designed to process an organization’s transactions and facilitate integrated real-time planning, production and customer response” (O’Leary, 2000, p. 27). David, Dunn, and McCarthy (1999) propose the use of the resources-events-agents (REA) enterprise ontology as a basis for comparison among systems and ERP packages. We agree REA is a robust candidate to which ERP systems may be compared because of its strong semantic, microeconomic, transaction and accounting heritage. More importantly, semantic models must be used as a basis for the information system because of the information contained within the semantics.<sup>1</sup>

The REA framework as an enterprise ontology provides a high-level definition and categorization of business concepts and rules, enterprise logic, and accounting conventions of independent and related organizations (Geerts & McCarthy, 2001b, 2003). The REA ontology includes three levels: the value chain level, the process level, and the task level. The value chain level models an enterprise’s “script” for doing business. That is, it identifies the high-level business processes or cycles<sup>2</sup> (e.g., revenue, acquisition, conversion, financing, etc.) in the enterprise’s value chain and the resource flows between those processes. The process level represents the semantic components of each business process. The task (or workflow) level of the REA ontology is the most detailed level and includes a breakdown of all steps necessary for the enterprise to accomplish the business events that were included at the process level.

The robust nature of the basic REA model has been demonstrated in research using various notations and implementation tools (Geerts & McCarthy, 1991; Nakamura & Johnson, 1998). What began as a simple model to capture semantics of accounting transactions more effectively than traditional double-entry techniques progressed to become an enterprise ontology (Geerts & McCarthy, 2001, 2003) and has contributed to inter-enterprise integration attempts such as the development of ebXML standards (e.g., RosettaNet, UNEDIFACT).

## BACKGROUND

The core REA model for each transaction cycle consists of the following components, presented in list form for brevity's sake. Readers are encouraged to read McCarthy (1982) for more detail.

- Two or more *economic events* that represent alternative sides of an economic exchange (at least one increment event and at least one decrement event).
- Two or more *resources* that represent what is received and given up in the economic exchange.
- *Internal agents* that represent the company's personnel that are responsible for each of the economic events (at least one agent for each event).
- One *external agent* that represents the person or company with whom the company is engaging "at arms' length" in the exchange.
- *Duality* relationship between the increment and decrement economic events.
- *Stock-flow* relationships between the events and the associated resources, representing the inflows or outflows of the resources resulting from the events.
- *Responsibility* relationships between the events and the internal agents.
- *Participation* relationships between the events and the external agents.

The following components were added in the progression from accounting model to enterprise ontology, again presented in list form. Readers are encouraged to read all of the Geerts and McCarthy articles listed in the references; David, Gerard, and McCarthy (2002); and Dunn et al. (2005) for more detail.

- Separation of the ontology into operational and knowledge (planning and control) levels to facilitate budgeting and management.
- Integration of transaction cycle models into an enterprise-wide value chain model.
- Expansion of transaction cycle models into workflow or task level models.
- Separation of components into *continuants* (enduring objects with stable attributes that allow them to be recognized on different occasions throughout a period of time) and *occurents* (processes or events that are in a state of flux).
- *Type images* that represent category-level abstractions of similar components.
- *Commitment images* that represent agreements to engage in future economic events.
- *Assignment* relationships between agents that represent the designation of an agent category to work

with another agent category (e.g., salesperson assigned to customer).

- *Custody* relationships between agents and resources that represent the agents that are accountable for various resources.
- *Fulfillment* relationships between commitment images and the resulting economic events.
- *Partner* relationships between commitment images and the participating agents.
- *Reservation* relationships between commitment images and the resources that are the proposed subject of the future exchange.
- *Typification description* relationships between continuant components and the categories to which they belong, e.g., resource-resource type and agent-agent type relationships.
- *Characterization description* relationships between continuant type images, e.g., agent type-agent type and agent type-resource type relationships.
- *Typification history* relationships between physical occurents and their types, indicating that the occurents share the same script, e.g., event-event type.
- *Scenario history* relationships between abstract occurents and other abstractions, e.g., event type-resource type.
- *Business process* as a description of the interaction between resources, agents, and dual events.
- *Partnering* as the purpose of the interaction between resources, agents, and commitments.
- *Segmentation* as a description of the grouping of physical categories into abstract continuant categories.
- *Policy or standard* as a description of the expression of knowledge-level rules between abstract types (e.g., scripts and scenarios).
- *Plan* as a description of the application of a script to physical occurents.
- *Strategy* as a description of rules for the execution of a business process or partnering.

## REA ONTOLOGY RESEARCH

Much attention has been given to the REA ontology in the literature. Published research papers have included design science and empirical methodologies. Textbooks include extensive coverage of the REA ontology. In this section we provide brief overviews of each of these categories and identify sources for interested readers.

Design science research associated with semantically modeled enterprise databases began before the advent of the REA ontology, in both computer science and account-



ing. As REA was developed, ideas were incorporated from scholars such as Codd (1970), Chen (1976), and Smith and Smith (1977). David et al. (2002) provide a detailed account of the design science research that influenced REA from its inception and throughout its subsequent progression. David et al. (2002) also details the research studies that have made significant design science advances in REA, separating them into those that created new constructs, models, methods, and instantiations.

Empirical research associated with semantically modeled enterprise databases investigates both enterprise- and individual-level benefits. Dunn and Grabski (2002) provide a detailed review of empirical research at both levels. Studies such as Andros, Cherrington, and Denna (1992) and Cherrington et al. (1996) revealed significant benefits from a semantically modeled system based on the REA model, including reductions in cost and processing time and increases in employee satisfaction. David (1995) observed productivity and administrative efficiencies for systems identified as more REA-like. Research by Weber (1986) and O'Leary (2004) compared the REA semantic model to existing software to determine consistencies and differences. Weber (1986) compared REA to wholesale distribution software; O'Leary (2004) compared REA to SAP, a leading ERP software package. Both studies found high-level consistency between the software and REA but found lower-level divergences due to accounting artifacts embedded in the software.

Dunn and Grabski (2002) reviewed individual-level research and semantically modeled enterprise databases and provided the following conclusions. Conceptual modeling formalisms are superior to logical modeling formalisms for design accuracy (Sinha & Vessey, 1999; Kim & March, 1995). The focus on increment and decrement resources and events along with the associated agents to those events is consistent with database designers' thought processes. Additionally, knowledge structures consistent with the REA template's structuring orientation are associated with more accurate conceptual accounting database design (controlling for knowledge content, ability, and experience level; Gerard, 1998). The lack of mandatory properties with entities is not critical (Bodart et al., 2001), perhaps because of the semantics inherent in the modeled system. System designers distinguish between entities and relationships, with entities being primary (Weber, 1996). Accounting systems based on the REA model are perceived as more semantically expressive by end users than are accounting systems based on the traditional debit-credit-account mode, and systems perceived as semantically expressive result in greater accuracy and satisfaction by end users than do non-semantically expressive systems (Dunn & Grabski, 2000). The ability to dis-embed the essential objects and relationships between the objects in complex surroundings de-

pends on a cognitive personality trait and field independence and leads to more accurate conceptual model design (at least for undergraduate students; Dunn & Grabski, 1998). Data and process methodologies are easier for novices than the object methodology and resulted in less unresolved difficulties during problem-solving processes (Vessey & Conger, 1994), and there is a more pronounced effect for process-oriented tasks (Agarwal, Sinha, & Tanniru, 1996a). Experience in process modeling matters, regardless of whether the modeling tool (process versus object-oriented) is consistent with the experience (Agarwal, Sinha, & Tanniru, 1996b).

REA is used to teach semantically modeled database design in many accounting and management information systems courses around the world. Several information systems textbooks have incorporated REA. Some cover REA in one or more chapters but also cover a wide variety of other systems topics (e.g., Gelinias, Sutton, & Oram, 2001; Hall, 2003; Romney & Steinbart, 2003). Others focus exclusively on REA (e.g., Dunn et al., 2005). The increasing coverage of REA in academic textbooks demonstrates its acceptance in the academic community.

## FUTURE TRENDS

Electronic commerce is a means of conducting business whereby electronic technologies facilitate communications between trading partners and enhance the value of their business relationships. The most common current use of information and communication technologies (i.e., information systems) for electronic commerce is to transmit data back and forth between companies in rigidly structured formats. In business-to-consumer electronic commerce, businesses create Web sites from which they display their product offerings to potential customers. Customers complete online order forms that transmit the data in the company's chosen format. Most business-to-business electronic commerce utilizes electronic data interchange with its rigid standards. Information systems structured in these ways meet the definition of conduits—they are pipes through which the data flows (and how well the data flows depends on the size and layout of the pipes). While they certainly add value beyond not having Internet Web sites or EDI or any other alternative, whether they maximize value is unclear. Use of a semantically based ontological modeling approach such as the REA ontology has the potential to enhance electronic commerce, particularly in the business-to-business environment.

Supply chain management issues are similar to those of e-commerce. Use of a semantically based ontological system like REA is a necessary but not sufficient condition to facilitate effective and efficient interorganizational

system integration. Intelligent agents and automated intensional reasoning are also required to facilitate such integration, and the system semantics must not be obscured by subsequent implementation artifacts.

Practices associated with data warehouses can benefit from semantically modeled systems. Data warehouses and data marts are used in most large firms today because of the current limitations of hardware and software. It is not physically possible to store all event data on a single system for current transaction processing and all system queries to obtain sufficient performance. However, on a conceptual level, a system designed based on the REA ontology negates the need for a separate data warehouse. Information in data warehouses is simply a set of “views” of the operational data prepared for specific functions. These views could be produced directly from the operational database and manipulated by users. Only technological speed, capacity, and security issues limit the feasibility of discarding data warehouses.

Many issues still need to be resolved, and as such these present many research opportunities. One issue focuses on the scalability of systems based on the REA ontology to support very small enterprises, large multinational firms, and firms of all sizes in between. Additional research is needed to build on automated intensional reasoning (Rockwell & McCarthy, 1999) and extend it to include the use of intelligent agents and object-based environments. Preservation of the semantics at an operational level, beyond that of the database itself, would allow decision makers additional insight into the problems and the information available to address the issues that they face. Again, object-based systems seem to provide the most benefit, but additional research is needed.

## CONCLUSION

Semantically modeled enterprise databases have demonstrated benefits for individuals, for enterprises, and for inter-enterprise system integration. To take full advantage of the semantically rich ontological patterns and templates, the REA ontology must be implemented with current advances in artificial intelligence technology and object-oriented database technology. Many current problems faced by companies who attempt to install ERP systems and integration tools such as EDI can be minimized by use of common semantic patterns about which intelligent systems can reason. Because REA systems are ontologically driven, they allow enterprises whose business practices are different from each other to realize their base constructs are the same and can be used for the basis of integration.

## REFERENCES

- Agarwal, R., Sinha, A. P., & Tanniru, M. (1996a). Cognitive fit in requirements modeling: A study of object and process methodologies. *Journal of Management Information Systems*, 13(2), 137-162.
- Agarwal, R., Sinha, A. P., & Tanniru, M. (1996b). The role of prior experience and task characteristics in object-oriented modeling: An empirical study. *International Journal of Human-Computer Studies*, 45, 639-667.
- Andros, D., Cherrington, J. O., & Denna, E. L. (1992, July/August). Reengineer your accounting the IBM way. *The Financial Executive*, 8(4), 28-31.
- Bodart, F., Patel, A., Sim, M., & Weber, R. (2001). Should optional properties be used in conceptual modeling? A theory and three empirical tests. *Information Systems Research*, 12(4), 384-405.
- Chen, P. P. (1976, March). The entity-relationship model—Toward a unified view of data. *ACM Transactions on Database Systems*, 1(1), 9-36.
- Cherrington, J. O., Denna, E. L., & Andros, D. P. (1996). Developing an event-based system: The case of IBM's national employee disbursement system. *Journal of Information Systems*, 10(1), 51-69.
- Codd, E. F. (1970, June). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6), 377-387.
- David, J. S. (1995). *An empirical analysis of REA accounting systems, productivity, and perceptions of competitive advantage*. Unpublished doctoral dissertation, East Lansing, MI: Michigan State University.
- David, J. S., Dunn, C. L., & McCarthy, W. E. (1999). *Enterprise resource planning systems research: The necessity of explicating and examining patterns in symbolic form*. East Lansing, MI: Michigan State University.
- David, J. S., Gerard, G., & McCarthy, W. E. (2002). Design science: An REA perspective on the future of AIS. In V. Arnold & S. Sutton (Eds.), *Researching accounting as an information systems discipline*. Sarasota, FL: American Accounting Association.
- Dunn, C. L., Cherrington, J. O., & Hollander, A. S. (2005). *Enterprise information systems: A pattern based approach* (3rd ed.). Burr Ridge, IL: McGraw-Hill Irwin.
- Dunn, C. L., & Grabski, S. V. (1998). The effect of field independence on conceptual modeling performance. *Advances in Accounting Information Systems*, 6, 65-77.

- Dunn, C. L., & Grabski, S. V. (2000). Perceived semantic expressiveness of accounting systems and the effect on task accuracy. *International Journal of Accounting Information Systems*, 1(2), 79-87.
- Dunn, C. L., & Grabski, S. V. (2002). Evaluative research in semantically modeled accounting systems. In V. Arnold & S. Sutton (Eds.), *Researching accounting as an information systems discipline*. Sarasota, FL: American Accounting Association.
- Dunn, C. L., & McCarthy, W. E. (1997). The REA accounting model: Intellectual heritage and prospects for progress. *Journal of Information Systems*, 11(1), 31-51.
- Geerts, G. L., & McCarthy, W. E. (1991). Database accounting systems. In B. C. Williams & B. J. Spaul (Eds.), *IT and accounting: The impact of information technology* (pp. 159-183). London: Chapman & Hall.
- Geerts, G. L., & McCarthy, W. E. (1998). *Accounting as romance: Patterns of unrequited love and incomplete exchanges in life and in business software*. Working paper presented at the Arizona State University REA Roundtable Workshop, Tempe, AZ.
- Geerts, G. L., & McCarthy, W. E. (1999, July/August). An accounting object infrastructure for knowledge-based enterprise models. *IEEE Intelligent Systems & Their Applications*, 14(4), 89-94.
- Geerts, G. L., & McCarthy, W. E. (2000). Augmented intensional reasoning in knowledge-based accounting systems. *Journal of Information Systems*, 14(2), 127-150.
- Geerts, G. L., & McCarthy, W. E. (2001a). Using object templates from the REA accounting model to engineer business processes and tasks. *The Review of Business Information Systems*, 5(4), 89-108.
- Geerts, G. L., & McCarthy, W. E. (2001b). *The ontological foundation of REA enterprise information systems* (Working paper). East Lansing, MI: Michigan State University.
- Geerts, G. L., & McCarthy, W. E. (2002). An ontological analysis of the economic primitives of the extended-REA enterprise information architecture. *International Journal of Accounting Information Systems*, 3(1), 1-16.
- Geerts, G. L., & McCarthy, W. E. (2003). *Type-level specification in REA enterprise systems* (Working paper). East Lansing, MI: Michigan State University.
- Gelinas, U. J., Sutton, S. G., & Oram, A. E. (2001). *Accounting information systems* (5th ed.). Mason, OH: South-Western College.
- Gerard, G. (1998). *REA knowledge acquisition and related conceptual database design performance*. Unpublished doctoral dissertation, East Lansing, MI: Michigan State University.
- Hall, J. (2003). *Accounting information systems* (4th ed.). Mason, OH: South-Western College.
- Kim, Y. K., & March, S. T. (1995). Comparing data modeling formalisms. *Communications of the ACM*, 38(6), 103-115.
- McCarthy, W. E. (1982). The REA accounting model: A generalized framework for accounting systems in a shared data environment. *The Accounting Review*, 57(3), 554-578.
- Nakamura, H., & Johnson, R. E. (1998). Adaptive framework for the REA accounting model. *Proceedings of the OOPSLA'98 Business Object Workshop IV*. Retrieved August 17, 2004, from <http://jeffsutherland.com/oopsla98/nakamura.html>
- O'Leary, D. E. (2000). *Enterprise resource planning systems: Systems, life cycle, electronic commerce, and risk*. New York: Cambridge University Press.
- O'Leary, D. E. (2004). On the relationship between REA and SAP. *International Journal of Accounting Information Systems*, 5(1), 65-81.
- Rockwell, S. R., & McCarthy, W. E. (1999). REACH: Automated database design integrating first-order theories, reconstructive expertise, and implementation heuristics for accounting information systems. *International Journal of Intelligent Systems in Accounting, Finance & Management*, 8(3), 181-197.
- Romney, M. B., & Steinbart, P. J. (2003). *Accounting information systems* (9th ed.). Upper Saddle River, NJ: Prentice Hall.
- Scheer, A.-W. (1998). *Business process engineering: Reference models for industrial enterprises*. Secaucus, NJ: Springer-Verlag New York, Inc.
- Sinha, A. P., & Vessey, I. (1999). An empirical investigation of entity-based and object-oriented data modeling. In P. De & J. DeGross (Eds.), *Proceedings of the Twentieth International Conference on Information Systems* (pp. 229-244). Charlotte, NC.
- Smith, J. M., & Smith, D. C. (1977, June). Database abstractions: Aggregation and generalization. *ACM Transactions on Database Systems*, 2(2), 105-133.
- Vessey, I., & Conger, S. A. (1994). Requirements specification: Learning object, process, and data methodologies. *Communications of the ACM*, 37(5), 102-113.
- Weber, R. (1986). Data models research in accounting: An evaluation of wholesale distribution software. *The Accounting Review*, 61(3), 498-518.

Weber, R. (1996). Are attributes entities? A study of database designers' memory structures. *Information Systems Research*, 7(2), 137-162.

## KEY TERMS

**Business Process:** A term widely used in business to indicate anything from a single activity, such as printing a report, to a set of activities, such as an entire transaction cycle. In this paper, business process is used as a synonym of transaction cycle.

**Enterprise Resource Planning System:** An enterprise-wide group of software applications centered on an integrated database designed to support a business process view of the organization and to balance the supply and demand for its resources. This software has multiple modules that may include manufacturing, distribution, personnel, payroll, and financials and is considered to provide the necessary infrastructure for electronic commerce.

**Ontologically-Based Information System:** An information system based upon a domain ontology, whereby the ontology provides the semantics inherent within the system. These systems facilitate organizational productivity and interorganizational communication.

**Process Level Model:** A second-level model in the REA ontology that documents the semantic components of all the business process events.

**REA Enterprise Ontology:** A domain ontology that defines constructs common to all enterprises and demonstrates how those constructs may be used to design a semantically modeled enterprise database.

**Semantically Modeled Enterprise Database:** A database that is a reflection of the reality of the activities in which an enterprise engages and the resources and people involved in those activities. The semantics are present in the conceptual model but might not be readily apparent in the implemented database.

**Task Level Model:** A third-level model in the REA ontology is the most detailed level, which specifies all steps necessary for the enterprise to accomplish the business events that were included at the process level.

**Value Chain:** The interconnection of business processes via resources that flow between them, with value being added to the resources as they flow from one process to the next.

## ENDNOTES

- <sup>1</sup> An alternative semantic model has been presented by Scheer (1998). Additional research is needed comparing these two semantic models with subsequent evaluations of commercially available ERP packages.
- <sup>2</sup> The REA framework uses the term business process to mean a set of related business events and other activities that are intended to accomplish a strategic objective of an organization. In this view, business processes represent a high level of abstraction. Some non-REA views define business processes as singular activities that are performed within a business. For example, some views consider "process sale order" to be a business process. The REA view considers "sale order" to be a business event that is made up of specific tasks, and it interacts with other business events within the "sales-collection" business process.



# Semistructured Data and its Conceptual Models

**Murali Mani**

*Worcester Polytechnic Institute, USA*

**Antonio Badia**

*University of Louisville, USA*

## INTRODUCTION

Semistructured data is data with no predefined schema, or a very flexible schema. Because such data does not fit well in traditional databases, new data models have been developed to deal with it. XML is the most well known of these new data models.

Traditional database systems have adapted to handle XML data by extending data types, query languages, indexing methods, and optimization techniques to the nature of XML; also, brand new database systems have been developed. However, the first step in developing a database is to create a conceptual model that can be used as the starting point for the design process (Davis, 1993). Despite the fact that existing conceptual models, most notably, Entity-Relationship (E-R) models (Chen, 1976), are inadequate for semistructured data because relatively little research is devoted to adapting conceptual models to the characteristics of this type of data (Badia, 2002; Elmasri et al., 2002). In this paper, we will review the main characteristics of XML and E-R, show the mismatch between them, and propose extensions to E-R that address those mismatches.

## BACKGROUND

In this section, we review the basic ideas of conceptual models and XML in order to fix some vocabulary for the rest of the paper.

## CONCEPTUAL MODELS

We briefly review the main characteristics of E-R models (Chen, 1976; Thalheim, 2000). There are other conceptual models, like Unified Modeling Language (UML) (Rumbaugh, Jacobson & Booch, 1999) and Object-Role-Modeling (ORM) (Halpin & Bloesch, 1999); we will not include them here for lack of space, although we will make a few comments later on.

An Entity-Relationship (E-R) (Chen, 1976; Thalheim, 2000) model is based on three basic concepts: *entity types*,

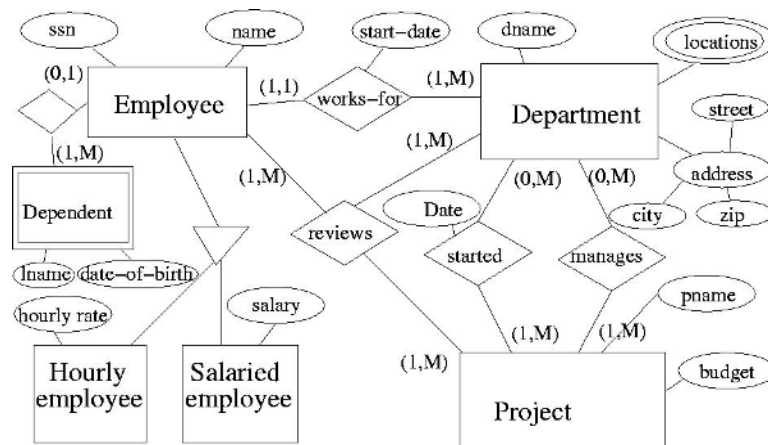
*attributes*, and *relationships*. E-R models are usually depicted in *E-R diagrams*; an example is given in Figure 1. Entity types are depicted as rectangles with a name inside, attributes as ovals, and relationships as lines with a diamond shape on them.

Entity types represent things either real or conceptual. They denote sets of objects, not particular objects; in this respect, they are close to *classes* in object-oriented models. The set of objects modeled by an entity type are called its *extension*. Particular objects are called *entities*.

Relationships are connections among entity types. Relationships may involve any number of entity types; those involving two entity types (called *binary relationships*) are the most common. However, n-ary relationships (involving  $n > 2$  entity types) are also possible. In particular, relationships relating one entity to itself are allowed. For example, a relationship Manager-of may relate the entity Employee to itself. To distinguish the ways in which Employee participates in this relationship, *roles* are added to the entity (*manager* and *managee*, in this example). Relationships are fundamental in an E-R model in that they carry very important information in the form of *constraints*: *participation constraint* tells us whether all objects in the extension of an entity are involved in the relationship or whether some may not be. For example, entities Department and Employee have a relationship works-for between them. If all employees work for some department, then participation of Employee in Works-for is total. However, if there can be employees which are not assigned to a particular department, then participation is partial. *Cardinality constraint* tells us how many times an object in the entity's extension may be involved in a relationship and allows us to classify binary relationship as *one-to-one*, *one-to-many*, or *many-to-many*. There are several notations to state constraints in an E-R diagram. The one chosen here associates with each entity type *E* and relationship *R* a pair of numbers (*min*, *max*), where *min* represents the minimum number of times an entity in *E* appears in *R* (thus, *min* represents the participation constraint by being 0 for partial and 1 for total), and *max* represents the maximum number of times an entity in *E* appears in *R* (thus, *max* represents the cardinality constraint by being 1 for one-to relationships and greater than



Figure 1. Example E-R schema



1 for many-to relationships. The latter case is traditionally represented by using the letters *M* or *N*. Thus, the (1, 1) by Employee and Works-for indicates that all employees work for exactly one department; the (0, *M*) by Department and Manages indicates that not all departments manage projects, but those that do may manage more than one, and so on.

Entity types and relationships may have *attributes*, which are properties with a *value*. Attributes convey characteristics or descriptive information about the entity type or relationship to which they belong. Attributes may be simple or composite (made up of simpler parts, like the attribute Address of entity Department in the example, which is made up of parts named street, city, and zip), single or multivalued (capable of having one or several values for a particular entity; multivalued attributes are displayed as dual ovals, like locations in the example above, meaning that some department may have multiple locations), and primitive or derived (a derived attribute value is computable from other information in the model). A *key attribute* is an attribute whose value is guaranteed to exist and be different for each entity in the entity type. Therefore, the attribute is enough to point out a particular entity. All entity types are assumed to have at least one key attribute.

A contentious issue is the semantics of entity type's attributes, in particular, whether attributes are *required* (i.e., every entity of the type must have a value for each attribute of the type) or *optional* (i.e., some entities of the type may or may not have values for some attributes). Different authors take different views on this issue, some even arguing that it is a mistake to consider attributes optional (Bodart et al., 2001). Since this has an impact

when transforming E-R models into different data models, we will point out how to deal with each view.

Some E-R models admit *weak entities*, entities with no key attributes; these entities are connected by a one-to-many relationship to a regular entity, called the *strong entity*. What characterizes a weak entity is its lack of clear identity (reflected in the lack of a key) and its dependence for existence on the strong entity. As a typical example, an entity loan may have an associated weak entity payment. Clearly, a loan is associated with several payments (hence, the one-to-many relationship), and if a loan ceases to exist (say it is paid off), then the associated payments also cease to exist.

Many proposals for additional features have been made over the years. The most successful one is the addition of *class hierarchies* by introducing IS-A (class/subclass) relations between entities. This addition, obviously motivated by the success of object-oriented methods for analysis, allows the designer to recognize commonalities among entities; usually this means shared attributes exist. Shared attributes are removed and put together in a new entity (class) which is a generalization of the others, and a class-subclass relationship is created. As in object-oriented approaches, *inheritance of attributes* is assumed. In Figure 1, entity type Employee has two subtypes, Hourly-employee and Salaried-Employee. The IS-A relationship is indicated by a downward triangle instead of a diamond in the line joining the involved entity types. The IS-A relationship can be annotated to distinguish several situations: whether the subclasses are disjoint or not and whether the subclasses together *cover* the superclass (i.e., every entity of the superclass must also belong to one of the subclasses) or not. Note that

both dimensions are orthogonal to each other; hence, two annotations are needed to determine the exact situation.

### XML

We briefly review the main characteristic of XML. Conceptually, we can see schema for an XML document as a (annotated) regular tree grammar with production rules of the form  $A \rightarrow aX$ , where  $A$  is a nonterminal symbol, “ $a$ ” is a terminal symbol, and  $X$  is a regular expression over the alphabet of nonterminal symbols. In a more descriptive analysis, XML describes elements (which are denoted by matching opening and closing tags) which may contain other elements and/or have attributes. XML has the ability to specify if a subelement may appear: exactly once (*regular* subelements); once or none at all (*optional* subelements); once or more times (*repeated* subelements); none, once or several times (*Kleene star* subelements), and whether two or more subelements may appear in a given position (*choice* subelements). Besides subelements, XML can also specify the attributes of an element. Attributes are properties with a (unique) value. All attributes of an element must be different, may have a default value, and be mandatory or optional. Because an element may contain itself, recursive definitions are allowed (i.e., part-subpart definitions). In XML Schema, a set of subelements and/or attributes of an element can be declared a key for that element; that key can then be referenced by other elements through the use of foreign keys.

There is much more to XML and XML Schema; here we are only interested in the main structural details. The interested reader is referred to the W3C standards (Bray, Paoli & Sperberg-McQueen, 2004).

### MODELING XML IN E-R

There is a straightforward translation of E-R into XML Schema that simply imitates the way E-R is translated into a relational design<sup>1</sup>. While this translation (made possible by the ability of XML Schema to specify integrity constraints) is technically correct, it is contrary to the way XML is intended to be used. In particular, there are several XML features that are of no relevance for such a translation. In other words, if we take an E-R model and transform it to an XML Schema in the manner just indicated, there are certain features that will never be used. One of them is the ability to form union types, another one is the ability to have recursive types, and a third one is the possibility of having order in relationships. Finally, there is also the issue of whether the ability to specify how many times a subelement is present can be used. The flat translation proposed denies the need to have complex subelements,

and normalization gets rid of repeated subelements (Date, 2000). There is the question of whether optional simple attributes would ever be used; this is a somewhat ambiguous question because of the ambiguity pointed out earlier in E-R semantics. If attributes are considered optional, then if we choose to represent E-R attributes as XML simple elements, we should consider them optional (if we translate into XML attributes, the #OPTIONAL or #REQUIRED keyword can be used for the same purposes).

The ability to have union types provides important flexibility in modeling. It allows us to recognize commonalities in function, for instance, even if structurally two elements are different. Assume, as an example, an entity *Journal\_Paper* and another entity *Conference\_Paper*. Both can be publications, therefore, be in a certain relationship *Written\_by* with entity *Author*. If this relationship is important, we would like to be able to see *Journal\_Paper* and *Conference\_Paper* as part of the same type (*Publication*) despite other differences. This is the root of the proposal to create categories in E-R models (Elmasri, Weeldreyer & Hevner, 1985). Categories are basically union entity types; when added to E-R models, they correspond to union types in XML. Following our example, *Publication* is a category, composed on entities *Journal\_Paper* and *Conference\_Paper*, and would get translated as an element embodying the union of such elements. The importance of categories is that they allow the expression of a new set of constraints, which we can call covering and overlap constraints. A covering constraint holds if every entity in a category belongs to one of the types making up the category (i.e., if all publications are either a journal paper or a conference paper). Otherwise, a non-covering constraint holds (i.e., some publications are neither journal papers nor conference papers). An overlap constraint holds if the types making up the category have empty intersections (i.e., if no paper can be at the same time a journal and a conference paper); otherwise, a non-overlap constraint holds. Note, by the way, that the same type of constraints may hold of the roles that an entity plays in relationships. Assume, for example, that entity *Person* is related to category *Publication* by relationship *Writes* and that the category is made up as before. We may then want to express the fact that all persons that are authors are the authors of journal or conference papers, nothing else (a covering constraint), or that no person is an author of both journal and conference papers (an overlap constraint).

The ability to have recursive types also gives the capability to model recursive relations like part and subpart. Finally, ordered relationships can be “simulated” in E-R models by adding an attribute *Order* to the relationship. However, it is still up to the analyst/designer to adhere to the intended semantics, as there is nothing in the E-R model that enforces the ordering.

Assume, for instance, entities Person and Book, and a relationship Author between them. The relationship is many-to-many, as some books have several authors, and some authors have written several books. However, we would like the relationship to be ordered since it may be important for an application to know which one is the first book of an author, the second, and so forth. Note that the order is implicit and enforced on the XML side; hence, no special notation is needed to signal it.

## TRANSLATING THE EXTENDED MODEL

Assume that we have an extended E-R model in which, besides the usual E-R constructs (entity types, relationships, attributes, relationship constraints), we also have categories, covering and overlap constraints (on categories and roles), recursive relationships, and ordered relationships. How would such a model get translated into XML? Here we offer a translation, including several options that are up to the translator:

- For each entity type E, create an element  $\text{tr}(E)$ ;
- For each attribute in the entity type, create an attribute or a simple subelement;
- For each one-to-many relationship R between entity types E1 (many side) and E2 (1 side),
  - Option 1: make  $\text{tr}(E2)$  a subelement of  $\text{tr}(E1)$ ;
  - Option 2: add an ID attribute to the  $\text{tr}(E1)$ , and IDREF attribute to  $\text{tr}(E2)$ ;
  - Option 3: add a primary key to  $\text{tr}(E1)$ , and a foreign key to  $\text{tr}(E2)$ ;
  - On each case, add all attributes of R as attributes or subelements of  $\text{tr}(E2)$  too.
- For each many-to-many relationship R between entity types E1 and E2,
  - Create a new element E; use IDs and IDREFs or key and primary keys to establish relationships between E and  $\text{tr}(E1)$  and E and  $\text{tr}(E2)$ ;
  - Add all attributes of R as attributes or simple subelements of E.
- For any relationship R involving entity types  $E1, \dots, E_n$ 
  - Create a new element E; use IDs and IDREFs or key and primary keys to establish binary relationships between E and  $\text{tr}(E1)$ , E and  $\text{tr}(E2)$ , ..., E and  $\text{tr}(E_n)$ ;

- Add all attributes of R as attributes or simple subelements of E.
- For each recursive relationship R on entity type E,
  - Add the name of  $\text{tr}(E)$  as a subelement of  $\text{tr}(E)$  (repeated subelement, if R is one-to-many).
- For each category C, made up of entity types E1 and E2,
  - Create an element  $\text{tr}(C)$ , defined as  $(\text{tr}(E1)|\text{tr}(E2))$  (i.e., a choice).

Some comments are in order about this translation. When dealing with one-to-many relationships, several options are offered. Observe that if one chooses to represent such relationship using subelements and the entity type being represented as a subelement is involved in other relationships, redundancy may occur. Assume, for instance, entity types Employee, Department, and Project such that relationship Works-for is one-to-many between Employee and Department, and Manages is one-to-many between Employee and Project. We can declare Employee a subelement of Department, or a subelement of Project, but should not do both. If we choose to declare Employee a subelement of Department, Option 2 or Option 3 should be used to model Manages (Lee, Mani & Chu, 2003).

As for many-to-many relationships, no options are offered. Such relationships are modeled by reification, that is, converting the relationship into an element as if it were another entity type. This is similar to what happens when E-R models are captured in ODL (Catell, 1997) and is the only acceptable solution to avoid redundancy and possible inconsistencies. Finally, for n-ary relations ( $n > 2$ ), the solution adopted is similar. However, it could be possible to produce more stylized designs by analyzing the relationship. The solution proposed is correct, but other options are also available. For instance, consider a ternary relationship Supplies between entity types Company, City, and Product. Then we can create elements Company, Product, City, and element Supply such that Supply is a subelement of Company, and it contains IDREFS or foreign keys for the IDs or primary keys for both Product and City (Mani, Lee & Muntz, 2001).

As for recursive relationships, note that if a relationship R on entity type E is not recursive (as in our example of relationship Married on entity type Person), one should treat it as a binary relationship and literally follow the rules for binary relationships. Note that there is no need to do anything about ordered relationships. Order is an implicit part of the XML data model.

With respect to categories, two points are important. First, by default, the translation given assumes that the

category is covering and non-overlapping; when this is not the case, the translation must be made a bit more complex. If the category is non-covering, then the union itself has to be considered part of a union. Assume a category Owner, made up of entities Bank and Person, and assume not all owners are banks or persons. The owner can be modeled as an element Owner with a union (Bank | Person | Other). Note that element Other may have no attributes or subelements. If the category is overlapping, a third choice must be given for overlapping objects that has all the attributes of the other elements. Assume a category Person which is made up of entity types Student and Staff; the category is overlapping (that is, some persons are both students and staff). Let  $\text{attr}(E)$  be the attributes of entity type  $E$ . Then Person can be represented as the union of three subelements, Student (with attributes  $\text{att}(\text{Student})$ ), Staff (with attributes  $\text{attr}(\text{Staff})$ ), and Student-Staff (with attributes  $\text{attr}(\text{Student}) \cup \text{attr}(\text{Staff})$ ). Second, some further processing is also possible. If  $C$  is both a covering and non-overlapping category, it is possible to dispense with creating elements for  $E1$  and  $E2$ . Instead, if  $\text{attr}(E1)$  denotes all subelements of  $E1$  and  $\text{attr}(E2)$  all subelements of  $E2$ , we can create an element  $\text{tr}(C)$  and define it as  $(\text{attr}(E1) | \text{attr}(E2))$ . When neither  $E1$  nor  $E2$  are involved in any relationships, this is all that is needed. If either one is involved in a relationship, we need to make sure the  $\text{tr}(E1)$  (or  $\text{tr}(E2)$ ) have either an ID or a primary key that can be used to represent the relationship.

## FUTURE TRENDS

We have emphasized here notions that seem necessary to model semistructured (XML) data; in particular, it seems clear that the idea of category and the expression of set constraints are needed if we are to use all the power of XML. Future research will have to focus on the issue of adding such constructs to conceptual models, as well as examining the need to develop specific ways to capture the irregular nature of semistructured data.

## CONCLUSION

It must be pointed out that some authors have argued that other data models are more suited than E-R for representing semistructured data (see Bird, Goodchild & Halpin, 2000; Conrad, Scheffner & Freytag, 2000). However, as far as the authors are aware, neither UML nor ORM can represent the set constraints (such as covering and overlap constraints) discussed in this entry, although they seem to be able to represent more constraints than E-R models. Given this, we believe that we need to propose

extensions to UML and ORM similar to that of covering and overlap constraints before we can use them as a conceptual model for XML. Also, since UML and ORM lack a formal semantics, it is impossible to ascertain with any certainty what the models can and cannot do. On the other hand, E-R has a well defined formal semantics, which allows us to specify the type of constraints it is missing.

## REFERENCES

- Badia, A. (2002). Conceptual modeling for semistructured data. *Proceedings of the International Workshop on Data Semantics in Web Information Systems (DASWIS)*.
- Bird, L., Goodchild, A., & Halpin, T.A. (2000). Object role modeling and XML schema. *Proceedings of the 19th International Conference on Conceptual Modeling*.
- Bodart, F., Patel, A., Sim, M., & Weber, R. (2001). Should optional attributes be used in conceptual modeling? A theory and three empirical tests. *Information Systems Research, 12*(4), 384-405.
- Bray, T., Paoli, J., & Sperberg-McQueen, C.M. (Eds). (2004). Extensible markup language (XML) 1.0, W3C recommendation. Retrieved February 2, 2005, from <http://www.w3.org/TR/REC-xml-20001006>
- Catell, R. (Ed). (1997). *The object database standard: ODMG 2.0*. Morgan Kaufmann.
- Chen, P. (1976). The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems, 1*.
- Conrad, R., Scheffner, D., & Freytag, J.C. (2000). XML conceptual modeling using UML. *Proceedings of ER 2000* (pp. 558-571).
- Date, C. (2000). *An introduction to database systems*. Addison-Wesley.
- Davis, F. (1993). *Requirement specification: Objects, functions and states*. Prentice Hall.
- Elmasri, R., Weeldreyer, J.A., & Hevner, A.R. (1985). The category concept: An extension to the entity-relationship model. *International Journal on Data and Knowledge Engineering, 1*(1).
- Elmasri, R., Wu, S., Hojabri, B., Fu, J., & Li, Q. (2002). Conceptual modeling for customized XML schemas. *Proceedings of the ER'02 Conference*.
- Halpin, T.A., & Bloesch, A. (1999). Data modeling in UML and ORM: A comparison. *Journal of Database Management*.



Lee, D., Mani, M., & Chu, W.W. (2003). Schema conversion methods between XML and relational models. *Knowledge Transformation for the Semantic Web*, 1-17.

Mani, M., Lee, D., & Muntz, R.R. (2001). Semantic data modeling using XML schemas. *Proceedings of ER*(pp. 149-163).

Rumbaugh, J., Jacobson, I., & Booch, G. (1999). *The unified modeling language reference manual*. Addison-Wesley.

Thalheim, B. (2000). *Entity-relationship modeling*. Springer-Verlag.

## KEY TERMS

**Category:** Special type of entity that represents the union of two or more different entity types.

**Conceptual Model:** Semiformal framework (usually a language and a diagram notation) used to capture information about the structure and organization of things, properties, and relations in a fragment of the real world, called the domain, usually one of interest to a (software) system. The model represents the semantics of the domain to the system.

**Conceptual Modeling:** Starting point for database design that consists of producing a conceptual model.

**Covering Constraint:** Constraint that states that the entity types that compose a category, taken together, contain all the elements of the category.

**Entity-Relationship Model:** One of the most well known and widely used conceptual models. The main concepts in the ER model are entity types, relationships, and their attributes.

**Overlap Constraint:** Constraint that states that the entity types that compose a category can (or cannot) have elements in common.

**Semistructured Data:** Data that does not adjust to a predefined schema or that has a very flexible, possibly dynamic, schema.

**XML:** Extensible Markup Language, a sublanguage of SGML that provides a way to structure and format data, especially in textual form. The data is represented as XML documents; an XML document may have an optional schema associated with it. XML is considered a semistructured data model because a schema is not always necessary, and even if a schema is present, it could be quite flexible.

## ENDNOTE

- <sup>1</sup> This translation still leaves open one important question: whether to model attributes in E-R models as attributes in XML or as simple subelements. However, this decision is not relevant for the structural issues we are discussing.



# Sensors, Uncertainty Models, and Probabilistic Queries

Reynold Cheng

Purdue University, USA

Sunil Prabhakar

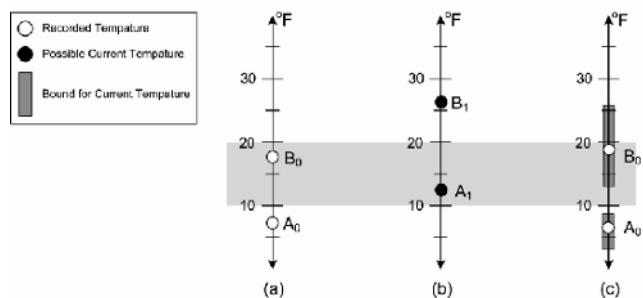
Purdue University, USA

## INTRODUCTION

Sensors are often used to monitor the status of an environment continuously. The sensor readings are reported to the application for making decisions and answering user queries. For example, a fire alarm system in a building employs temperature sensors to detect any abrupt change in temperature. An aircraft is equipped with sensors to track wind speed, and radars are used to report the aircraft's location to a military application. These applications usually include a database or server to which the sensor readings are sent. Limited network bandwidth and battery power imply that it is often not practical for the server to record the exact status of an entity it monitors at every time instant. In particular, if the value of an entity (e.g., temperature, location) monitored is constantly evolving, the recorded data value may differ from the actual value. Querying the database can then produce incorrect results. Consider a simple example where a user asks the database: "Which room has a temperature between 10°F and 20°F?" If we represent temperature values of rooms A and B stored in the database by  $A_0$  and  $B_0$  respectively, we can see from Figure 1(a) that the answer to the user query is "Room B". In reality, the temperature values of both rooms may have changed to newer values,  $A_1$  and  $B_1$ , as shown in Figure 1(b), where the true query answer should be "Room A". Unfortunately, because of transmission delay, these newest pieces of information are not propagated in time to the system to supply fresh data to the query, and consequently the query is unable to yield a correct answer.

In general, the incorrectness of query results is due to *sensor uncertainty*, an inherent property of any sensor database where each recorded data item is only an older, approximate version of the corresponding entity monitored. Apparently, providing meaningful answers in face of sensor uncertainty appears to be a futile exercise. In many situations, however, the values of sensors cannot change drastically in a short period of time – the degree and/or rate of change of a sensor value

Figure 1. Example of data uncertainty and a range query for temperature values



is constrained. This helps us to alleviate the problem. In the previous example, if we can guarantee that the actual temperatures of room A and B are no more than some deviations from  $A_0$  and  $B_0$  respectively, as in Figure 1(c), then we can state with confidence that room A does not satisfy the query.

Whether room B satisfies the query is less obvious. In Figure 1(c), it is not clear whether B has a temperature between 10°F and 20°F. However, the fact that the uncertainty associated with B's temperature is *bounded* makes it possible to decide the degree of likelihood that B satisfies this query. In general, bounded uncertainty allows us to augment different levels of confidence with each answer (e.g., as a probability), instead of providing a definite answer. Queries that augment answers with probability values, based on uncertain information, are known as *probabilistic queries*. In the previous example, a probabilistic query produces answers such as: "Room A has a probability of 0 being between 10°F and 20°F, while Room B has a probability of 0.7". Contrast with a query that gives incorrect answers because of stale data, a probabilistic query provides more confidence in the answers since uncertainty is considered.

Depending on the nature of the probabilistic query, confidence in a probabilistic query answer can be expressed in different forms. We will study different classes of probabilistic queries, which have different

forms of answers and evaluation techniques. We also examine the concept of “quality” of a probabilistic query result which is related to the ambiguity of the result.

The rest of this article is organized as follows. We first present the background and related works. Then we examine in detail how sensor data uncertainty can be modeled. Based on the data models, we present a classification of probabilistic queries. For each query class, we examine factors that determine quality of probabilistic query answers. Finally, we outline future research issues.

## BACKGROUND

Approximate answers to queries based on a subset of data have been well studied. Vrbsky and Liu (1994) studied approximate answers for set-valued queries (where a query answer contains a set of objects) and single-valued queries (where a query answer contains a single value). An exact answer  $E$  can be approximated by two sets: a certain set  $C$  which is the subset of  $E$ , and a possible set  $P$  such that  $C \cup P$  is a superset of  $E$ . Other techniques like precomputation (Poosala & Ganti, 1999), sampling (Gibbons & Matias, 1998), and synopses (Acharya, Gibbons, Poosala & Ramaswamy, 1999) are used to produce statistical results. While these efforts investigate approximate answers based on a subset of (exact) values of the data, this article addresses imprecise answers that assume all (uncertain) values of the data.

There are a number of works about evaluation of intervals. Olston et al. discuss the problem of balancing the trade-off between precision and performance for querying replicated data (Olston, Loo & Widom, 2001; Olston & Widom, 2000, 2002). In their model, the cache in the server cannot keep track of the exact values of sensor sources due to limited network bandwidth. Instead of storing the actual value in the server’s cache, an interval for each item is stored, within which the current value must be located. A query is then answered by using these intervals, together with the actual values fetched from the sources. The problem of minimizing the update cost within an error bound specified by aggregate queries is studied.

Probabilistic range querying over uncertainty of moving objects is presented by Wolfson, Sistla, Chamberlain, and Yesha (1999). Cheng, Kalashnikov, and Prabhakar (2004) present a solution for probabilistic nearest neighbor queries over moving objects. A generalized taxonomy of uncertainty models is presented by Cheng and Prabhakar (2003). Evaluation techniques and quality metrics are studied by Cheng, Kalashnikov, and Prabhakar (2003).

## SENSOR UNCERTAINTY MODELS

Uncertainty of sensor values can be represented in three forms, from no attention to uncertainty at all, to the highest resolution of uncertainty information (Cheng & Prabhakar, 2003). For notational convenience, let us assume that a real-valued attribute  $a$  of a set of database objects  $T$  is queried. We name the  $i$ th object of  $T$  as  $T_i$ , and the value of  $a$  for  $T_i$  as  $T_i.a$  (where  $i = 1, \dots, |T|$ ).

1. **Point Uncertainty:** This is the simplest model, where we assume there is no uncertainty associated with data at all. Each data item,  $T_i.a$ , is supposed to be a correct representation of the external entity being monitored. Queries use these exact values to evaluate results. Although manipulating real values is relatively easy for a query, the example in Figure 1 illustrates how such data can lead to incorrect query results.
2. **Interval Uncertainty:** Instead of representing the exact sensor value in the database, an *uncertain interval*, denoted by  $U_i(t)$ , is stored. Specifically,  $U_i(t)$  is a close interval  $[l_i(t), u_i(t)]$ , where  $l_i(t)$  and  $u_i(t)$  are real-valued functions of  $t$ , bounding the value of  $T_i.a$  at time  $t$ . This model represents imprecision of data in the form of an interval. An example model of  $U_i(t)$  is an interval bounding all values within a distance of  $(t - t_{update}) \times r$  of  $T_i.a$ , where  $t_{update}$  is the time that  $T_i.a$  is last updated, and  $r$  is the current rate of change of  $T_i.a$ . Thus,  $U_i(t)$  expands linearly with time until the next update of  $T_i.a$  is received. Another realization of this model can be found in Wolfson et al. (1999).
3. **Probabilistic Uncertainty:** This model is proposed by Cheng et al. (2003). Compared with interval uncertainty, it requires one more piece of information – the probability density function (pdf) of  $T_i.a$  within  $U_i(t)$ . We call this function an *uncertain pdf of  $T_i.a$*  at time  $t$ , denoted by  $f_i(x, t)$ . Notice that  $\int_{l_i(t)}^{u_i(t)} f_i(x, t) dx = 1$  and  $f_i(x, t)$  equals 0 if  $x \notin U_i(t)$ . Further, the exact form of  $f_i(x, t)$  is application-dependent. For example, in modeling sensor measurement uncertainty, where each  $U_i$  is an error range containing the mean value,  $f_i(x, t)$  can be a normal distribution around the mean value. Another example is the modeling of one-dimensional moving objects, where at any point in time, the actual location is within a certain bound,  $d$ , of its last reported location value. If the actual location changes further than  $d$ , then the sensor reports its new location value to the database and possibly changes  $d$ . Then  $U_i(t)$  contains all the values within

a distance of  $d$  from its last reported value (Wolfson et al., 1999), and  $f_i(x,t)$  can be a uniform distribution, that is,

$$f_i(x,t) = 1/[u_i(t) - l_i(t)] \text{ for } T_i.a \in U_i(t)$$

which models the scenario when  $T_i.a$  has an equal chance of locating anywhere in  $U_i(t)$ . Alternatively, one may perform an estimation of the pdf based on time-series analysis, the discussion of which is beyond the scope of this article, and interested readers are referred to Chatfield (1989) for details.

As we can see, the probabilistic uncertainty model is the most complicated model of sensor uncertainty among the ones we presented. The complexity of the model is paid off, however, by the fact that probabilistic queries can be defined, which we discuss in the next section.

## CLASSIFICATION OF PROBABILISTIC QUERIES

A *probabilistic query* assumes that data inputs are characterized under the probabilistic uncertainty model and augments the answers with confidence, expressed in probability values. In this section, we examine different types of probabilistic queries. In particular, we present a classification scheme of probabilistic queries proposed by Cheng et al. (2003).

Probabilistic queries can be classified in two ways.<sup>1</sup> First, we can classify them according to the forms of answers required. An *entity-based query* is one that returns a set of objects (e.g., list of objects that satisfy a range query), whereas a *value-based query* returns a single numeric value (e.g., value of a particular sensor). Another criterion is based on whether an aggregate operator, such as SUM and MAX, is used to produce results. An *aggregate query* is one where interplay between objects determines the results. For example, to decide whether an object holds the minimum value of  $T_i.a$  requires us to examine other objects as well. In contrast, for a *non-aggregate query*, the suitability of an object as the result to an answer is independent of other objects. A *range query* is a typical example—whether an object satisfies the range query is not affected by the values of other objects.

Based on this classification, we obtain four classes of probabilistic queries. For the names of the queries defined, the first letter is either *E* for entity-based query or *V* for value-based query.

1. **Value-Based Non-Aggregate Class:** This query class returns a single value for a given object as the

only answer and does not involve any aggregate operations. An example is the *probabilistic single value query* (VSingleQ), which returns the probabilistic uncertainty information (i.e.,  $U_i(t)$  and  $f_i(x,t)$ ) of a given object  $T_i$ .

2. **Entity-Based Non-aggregate Class:** This query class returns a set of objects, where the satisfiability of each object to the query is independent of others. One such query is the *probabilistic range query* (ERQ): given a closed interval  $[l,u]$ , a list of tuples  $(T_i,p_i)$  are returned, where  $p_i$  is the non-zero probability that  $T_i.a \in [l,u]$ .

3. **Entity-Based Aggregate Class:** In this query class, an aggregate operator is involved, and a set of objects together with their probability values, that is, a list of  $n$  tuples  $(T_i,p_i)$  is returned, with

$\sum_{i=1}^n p_i = 1$ . A typical example is the *probabilistic minimum query* (EMinQ): a set of tuples  $(T_i,p_i)$  are returned, where  $p_i$  is the non-zero probability that  $T_i.a$  is the minimum among all items in  $T$ .

4. **Value-Based Aggregate Class:** An aggregate operation is involved in deciding a single value to be returned. As an example, a *probabilistic sum query* (VSumQ) yields  $l,u \in \mathfrak{R}$  as well as  $\{p(x) \mid x \in [l,u]\}$ , where  $X$  is a random variable for the sum of values of  $a$  for all objects in  $T$ , and  $p(x)$  is

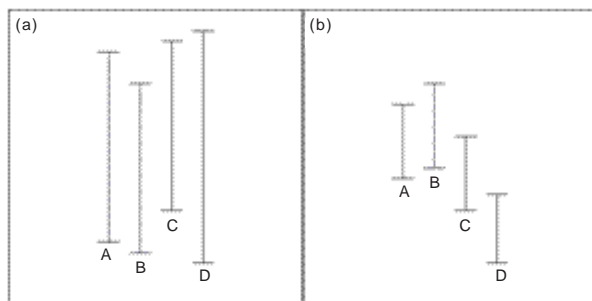
a pdf of  $X$  such that  $\int_l^u p(x)dx = 1$ .

This classification scheme allows us to develop evaluation algorithms for different types of probabilistic queries. Consider the evaluation of an ERQ. Since each object  $T_i$  can be evaluated independently of other objects, we first compute the overlapping interval *OI* of the two intervals,  $U_i(t)$  and  $[l,u]$ . The probability that  $T_i$  satisfies the ERQ, that is,  $p_i$ , is then calculated by integrating  $f_i(x,t)$  over *OI*. On the other hand, evaluating an EMinQ requires a completely different algorithm, details of which can be found in Cheng et al. (2003).

## QUALITY OF PROBABILISTIC RESULTS

As mentioned before, a probabilistic query returns answers augmented with probability values, as opposed to definite answers produced by a traditional query. A certain degree of imprecision is incorporated into the query answer, expressed in the form of probability values. The imprecision of a query answer is due to the fact that data supplied to the probabilistic query are inherently uncertain. In particular, if the data interested to a query contain a lot of uncertainty (e.g., their

Figure 2. Uncertainty and quality of results. In (a), it is not clear which item has a dominant probability value of satisfying EMinQ, and the quality of the result is poor. In (b), item D has a higher chance of satisfying EMinQ, yielding a higher quality result.



uncertain intervals are large), a probabilistic query can yield ambiguous answers which may not be very useful to the user. Figure 2(a) shows four items with large uncertain intervals that overlap each other. Assume these items have the same uncertain pdf. We can see that it is difficult to decide which item has a larger chance of being the minimum: the probability that satisfies the EMinQ is roughly the same for the four items. The quality of the result to this EMinQ is said to be “poor”, since it gives a vague answer and reflects that there is too much uncertainty for the query to yield a reasonable answer. On the contrary, the uncertainty intervals in Figure 2(b) are smaller, and apparently item D has a higher probability of having the minimum value, yielding a better quality for an EMinQ.

The quality of a probabilistic result is an important issue. It enables us to determine if the uncertainty of any sensor data needs to be reduced (e.g., request an immediate update from the sensor) in order to improve the quality of result. In this section, we study two key factors that affect quality – *size of uncertain intervals* and *entropy of probabilistic results*.

1. **The size of an uncertain interval** indicates the extent of error inherent to a data value. A larger uncertain interval can result in poorer answer quality, as illustrated in Figure 2. Another example is the evaluation of VSumQ over two uncertain data items  $T_i$  and  $T_j$ . The resulting value (i.e.,  $T_i.a + T_j.a$ ) lies in the error range  $[l_i(t) + l_j(t), u_i(t) + u_j(t)]$ . It is desirable to have smaller  $U_i(t)$  and  $U_j(t)$  in order to obtain a smaller error range, or higher answer quality.
2. **Entropy of probabilistic results:** Consider two sets of answers to an EMinQ:  $\{(T_1, 0.3), (T_2, 0.2), (T_3, 0.5)\}$  and  $\{(T_1, 0.8), (T_2, 0.2)\}$ . How do we know

which answer is more informative? Here, *entropy* gives us a convenient metric for measuring uncertainty of probabilistic results. According to Shannon (1949), entropy is defined as follows:

Let  $X_1, \dots, X_n$  be all possible messages, with respective probabilities  $p(X_1), \dots, p(X_n)$  such that  $\sum_{i=1}^n p(X_i) = 1$ . Then the entropy of a message  $X \in \{X_1, \dots, X_n\}$  is defined as

$$H(X) = \sum_{i=1}^n p(X_i) \log_2 \frac{1}{p(X_i)}.$$

Intuitively,  $H(X)$  measures the amount of uncertainty associated with a random message  $X$ . The larger the value of  $H(X)$ , the more amount of uncertainty is in  $X$ .

We can use the idea of entropy to quantify the quality of query answers. Consider the entity-based non-aggregate query class, where each object is evaluated independent of other queries. The quality of each answer tuple  $(T_i, p_i)$  can be specified in terms of entropy. For example, in an ERQ, each object  $T_i$  has a probability  $p_i$  of satisfying the query, and  $(1-p_i)$  otherwise. The entropy of  $(T_i, p_i)$  is then equal to  $-(p_i \log_2 p_i + (1-p_i) \log_2 (1-p_i))$ . The overall entropy of an ERQ result is then equal to the average of entropy values over all tuples of  $(T_i, p_i)$  where  $p_i \neq 0$ .

For entity-based aggregate queries, recall that a set  $R$  of tuples  $(T_i, p_i)$  are returned, with  $\sum_{i=1}^n p_i = 1$ . We can thus use  $H(R)$  to measure the answer uncertainty to these queries. The quality of the answer is lower if  $H(R)$  is higher.

A continuous version of the entropy definition can be used for value-based queries, defined as

$$\hat{H}(X) = \int_l^u p(x) \log_2 p(x) dx$$

where  $\hat{H}(X)$  is called the *differential entropy* of continuous random variable  $X$  with probability density function  $p(x)$  defined in the interval  $[l, u]$ .

For more details on answer quality metrics for different query classes and heuristics that improve query quality, readers are referred to Cheng et al. (2003).

## FUTURE TRENDS

Our next work is to study the indexing of probabilistic uncertain data in order to provide efficient access mechanisms to uncertain data. We also plan to study the join operation of uncertain data. We will also examine the efficient execution of a *probabilistic threshold query*, which is a probabilistic query with threshold  $\lambda$ —only



objects with probability value greater than  $\lambda$  are included in the answer. A preliminary study by Cheng & Prabhakar (2003) reveals that this variant of probabilistic queries is worth further study because it can improve the efficiency of probabilistic query algorithms significantly, by exploiting the fact that only objects with probability values higher than  $\lambda$  are returned.

## CONCLUSION

In this article, we studied how uncertainty information can be incorporated to data values. Under the probabilistic uncertainty model, we discussed probabilistic queries where confidence in the result is expressed in terms of probability values. Probabilistic queries can be classified into two dimensions, according to the nature of a query answer, as well as whether aggregation operations are involved. Each query class requires a different evaluation algorithm, as well as an answer quality metric. We also discuss how the sizes of uncertain intervals and entropy of the answer affect the quality of a probabilistic result.

## REFERENCES

- Acharya, S., Gibbons, P., Poosala, V., & Ramaswamy, S. (1999). Join synopses for approximate query answering. *Proceedings of the ACM SIGMOD International Conference on Management of Data*.
- Chatfield, C. (1989). *The analysis of time series an introduction*. Chapman and Hall.
- Cheng, R., Kalashnikov, D., & Prabhakar, S. (2003). Evaluating probabilistic queries over imprecise data. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, June.
- Cheng, R., Kalashnikov, D., & Prabhakar, S. (2004). Querying imprecise data in moving object environments. *IEEE Transactions on Knowledge and Data Engineering*, (to appear).
- Cheng, R., & Prabhakar, S. (2003, December). Managing uncertainty in sensor databases. *SIGMOD Record Issue on Sensor Technology*.
- Gibbons, P., & Matias, Y. (1998). New sampling-based summary statistics for improving approximate query answers. *Proceedings of the ACM SIGMOD International Conference on Management of Data*.

Olston, C., Loo, B.T., & Widom, J. (2001). Adaptive precision setting for cached approximate values. *Proceedings of the ACM SIGMOD International Conference on Management of Data*.

Olston, C., & Widom, J. (2000). Offering a precision-performance tradeoff for aggregation queries over replicated data. *Proceedings of the 26th International Conference on Very Large Data Bases*.

Olston, C., & Widom, J. (2002). Best-effort cache synchronization with source cooperation. *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 73-84).

Poosala, V., & Ganti, V. (1999). Fast approximate query answering using precomputed statistics. *Proceedings of the 15th International Conference on Data Engineering* (pp. 252).

Shannon, C. (1949). *The mathematical theory of communication*. University of Illinois Press.

Vrbsky, S.V., & Liu, J.W.S. (1994). Producing approximate answers to set- and single-valued queries. *The Journal of Systems and Software*, 27(3).

Wolfson, O., Sistla, P., Chamberlain, S., & Yesha, Y. (1999). Updating and querying databases that track mobile units. *Distributed and Parallel Databases*, 7(3).

## KEY TERMS

**Entity-Based Query:** A probabilistic query that returns a set of objects.

**Interval Uncertainty:** A model of sensor uncertainty where each stored data value is represented by a closed interval, called *uncertain interval*.

**Point Uncertainty:** A model of sensor uncertainty where each stored data item is assumed to be a correct representation of the entity being represented.

**Probabilistic Query:** A query which assumes data are characterized by probabilistic uncertainty, and returns query answers augmented with probability values.

**Probabilistic Threshold Query:** A probabilistic query with probability threshold  $\lambda$ , where only objects with probability values greater than  $\lambda$  are included in the answer.

**Probabilistic Uncertainty:** A model of sensor uncertainty, where each data value is represented by its *uncer-*



*tain interval*, together with a probabilistic density function (called *uncertain pdf*) that describes the distribution of the values within the interval.

**Sensor Uncertainty:** An inherent property of a sensor-based application, where each recorded data item in the database is only an older, approximate version of the entity being monitored in the external environment.

**Value-Based Query:** A probabilistic query that returns a single value.

## ENDNOTE

- <sup>1</sup> We only consider “atomic” queries that perform a single operation (e.g., range query). Complex queries that involve two or more operators (e.g., a SUM operation over a range query) are not covered by this classification scheme.

# Service Mechanism Quality for Enhanced Mobile Multimedia Database Query Processing

S

**Yanpu Zhang**

*University of Nebraska at Omaha, USA*

**Zhengxin Chen**

*University of Nebraska at Omaha, USA*

## INTRODUCTION

Among the challenges of multimedia and mobile computing, providing a mechanism for data retrieval in multimedia databases under wireless mobile environments is one of the most difficult issues (Shih, 2001). Up to now, the fundamental technologies that are specialized for wireless mobile, multimedia environments are not mature in object-oriented, object-relational, as well as relational databases (Hillborg, 2002; Ramakrishnan & Gehrke, 2003; Watson, 2004). An important issue is how to ensure quick query response for the users. If a user found out that the retrieved multimedia object is neither interesting nor useful after it is displayed, then the time and bandwidth used for transmitting the multimedia objects have already been wasted. In order to save precious time and expensive bandwidth, it could be a good idea to let users browse objects at an acceptable resolution without paying much attention to the details or at the limited device display capability. This article presents a novel concept to deal with this problem by making use of the concept of *quality of service* (QoS) to achieve adaptive query processing. In general, traditional QoS management is defined as the necessary supervision and control to ensure that the desired quality of service properties are attained and sustained, which applies both to continuous media interactions and to discrete interactions (Chalmers & Sloman, 1999). QoS thus consists of a set of specific requirements for

a particular service provided by a network to users. However, little work has been done in extending QoS principles to multimedia data management in wireless network environments.

## MOTIVATION

Since a traditional DBMS does not support QoS-based objects (Shih, 2002), it only concentrates on tables that contain a large number of tuples, each of which is of relatively small size. However, a multimedia database management system (MM-DBMS) should support QoS-sensitive multimedia data types in addition to providing all the facilities for DBMS functions. Once multimedia objects such as images, sound clips, and videos are stored in a database, individual objects of very large size have to be handled efficiently (Chang, Hung, & Shih, 2002). Furthermore, the crucial point is that mobile MM-DBMSs should have the QoS-based capabilities to efficiently and effectively process the multimedia data in wireless mobile environments.

Figure 1 depicts a scenario where a client degrades his quality criteria and obtains a coarse query result with his limited display capacity within the acceptable response time. Suppose the client with a currently available PDA device queries a University of Nebraska Peter Kiewit Institute (PKI) image in a multimedia database through wireless networks. The client uses a GPRS

Figure 1. Comparison of original and processed query image enhanced by extended QoS

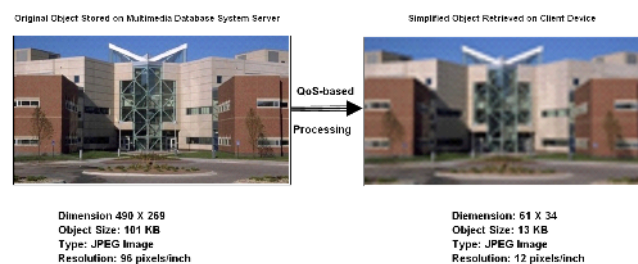
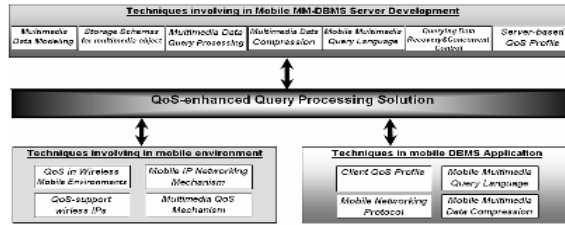


Figure 2. Technologies for QoS-based query processing in MM-DBMSs



system to connect to the UMTS wireless network. The server uses a wireless LAN (Wi-Fi) to connect to the wireless network. There are two problems existing in the client site: (1) The display capacity is too low to present the captured image object (the original dimension is 490×269 pixels, while the screen can only accommodate 128×128 pixels); (2) Even if the client can solve the screen resolution problem, the response time is still a problem due to the “out of the upper bound” delay time.

Using extended QoS-enhanced query processing in MM-DBMSs under wireless mobile environments, both issues mentioned above can be solved. The basic idea is that we reduce the final quality of picture to 12% of that of original one for the compensation, so that we can obtain the simplified object size from 101 KB to 12 KB. In this case, the data transmission time could be possibly controlled within 60 ms (since  $t_1 = 64\text{KB} / 512\text{KB} \times 50\% = 0.5 \text{ sec}$ ). However, such kind of adaptive QoS mechanism still does not exist.

## BACKGROUND

Numerous studies have been carried out for using QoS for mobile database query processing (Bordbar, Derrick, & Waters, 2002; Cao, 2003; Ecklund, Goebel, Plagemann, & Ecklund, 2002; Kazantzidis, 2002; Miloucheva & Tartarelli, 2002; Watson 2004). Note existing studies typically assume that all the QoS requirements are specified by users in advance. However, in multimedia applications, it is difficult to predict the size of the targeted object to be retrieved. Existing approaches usually stop the query if the required QoS conditions cannot meet the related statistical or empirical resource utilizations (Braumandl & Kemper, 2003). Apparently, stopping the query under an adverse condition is an unreasonable restriction, and adaptive query processing should be supported. Consequently, a critical issue to be investigated is concerned with how to extend existing QoS principles to deal with wireless mobile environments for multimedia applications. Technologies for QoS-based query processing in mobile, mul-

timedia DBMS are summarized in Figure 2. Extending query processing in MM-DBMSs to wireless mobile environment must consider the following: (1) Multimedia application, especially multimedia object retrieval, needs resource consumption often exceeding the available resource capacities of the deployed wireless/mobile networks and portable devices; (2) the precautions of these extra resource requirements cannot be taken by the server, the client, or even the network infrastructure in advance; and (3) we need to extend QoS management in a mobile environment to specify a range of acceptable QoS levels to allow for scaling of multimedia query processing, rather than trying to guarantee specific values or to stop the querying.

## A Quantitative Approach

Based on these considerations, we have explored a new *quantitative* approach to achieve the trade-off between querying results and qualities according to application priorities and capacities, because QoS requirements are preferably described by using some quantitative figures. Instead of requiring a good network service, the user is asked to specifically request some measures such as connection speed or delay, which can be described by a numerical value, for example, 256 Kbps or 60 ms, respectively, for the speed and delay. Having a quality term such as good or bad described by a quantitative metric simplifies the process of allocation of that quality to a particular service by the provider and also prevents any possible ambiguity during the user request and service fulfillment process (Ganz, Ganz, & Wongthavarawat, 2004). We have explored an approach to tackle this issue by specifying a range of acceptable QoS requirements for multimedia query processing. We have proposed a QoS-based matrix to support adaptive query processing of object-relational multimedia databases in the context of wireless mobile environments. The proposed QoS-based querying processing precision matrix (QQPPM) is based on real-time QoS conditions in wireless networks, the multimedia database’s object properties, and mobile client-site data processing

capability. In our research, we have focused on examining and establishing a server-site QoS conditions profile, wireless network QoS services profile, and client-site QoS requirements profile. Based on these QoS profiles, we have developed a QoS-based query processing precision matrix to approach this issue. The matrix representation is based on the following categorized characteristics.

- The wireless network QoS service profile is the real-time QoS for the particular application based on the conditions of on-site wireless networks;
- The server-site QoS profile is the properties of the multimedia object retrieved from the MM-DBMS server; and
- The client-site QoS profile is the mobile client's QoS requirements and client-site multimedia data processing capabilities.

The objective of constructing the QQPPM is to create a smallest element as a criterion to reduce queried multimedia object quality between the MM-DBMS server's original data quality and actual quality that client can access.

### Experiments

At the current time, there is neither a commercial MM-DBMS (Elmasri & Navathe, 2004) nor real wireless network facility that allows us to implement the QQPPM approach. Therefore, in order to verify the conception of QQPPM is reasonable and practicable, we have designed experiments focusing on examining the response time due to the fact that the limited bandwidth provided by wireless networks conflicts with the extraordinary resources required by a multimedia queried object's transmission. We have conducted simulations of transmitting different real-world multimedia data sizes with different infrastructure densities under wireless mobile environments. With proposed network QoS services, we have investigated the relationships among scaleable number of users, multimedia data sizes, corresponding systems' response times, and network traffics.

### CONCLUSION

We studied and extended the standard QoS mechanism in the context of wireless mobile environments. The client consideration elements such as hardware facilities, user preference, mobility coverage, and critical quality acceptance are included in the extended QoS framework. Rather than using statistical resource utilizations, we combined

with all client-site QoS considerations, server-site QoS requirements, and network QoS services into one simple matrix. Based on balance of all the QoS elements in the matrix, we calculated the precision criterion factor and applied it to process the queried object transmission. We conducted real-world wireless, multimedia database query processing simulations with different queried object sizes and different client densities on a traditional wireless network and a state-of-the-art wireless network and evaluated the relationship among object size, response time, and client densities.

We have reached the following results through this research:

- Multimedia object retrieval needs resource consumption that often exceeds available resource capacities of the deployed wireless mobile networks and portable devices.
- The precautions of extra resource requirements can not be taken by server, client, or even network infrastructure in advance. We need QoS management in wireless mobile environments to specify a range of acceptable QoS levels to allow for scaling of multimedia query processing, rather than trying either to guarantee specific values or to stop the querying.
- The model of QQPPM is reasonable and practicable. The theoretical studies and experimental practices support its possibility.

### REFERENCES

- Bordbar, B., Derrick, J., & Waters, G. (2002). Using UML to specify QoS constraints in ODP. *Computer Networks*, 40(2), 279-304.
- Braumandl, R., & Kemper, A. (2003). Quality of service in an information economy. *ACM Transactions on Internet Technology*, 3(4), 291-333.
- Cao, G. (2003). Integrating distributed channel allocation and adaptive handoff management for QoS-sensitive cellular networks. *Wireless Networks*, 9, 131-142.
- Chalmers, D., & Sloman, M. (1999). A survey of quality of service in mobile computing environments. *IEEE Communications Surveys*, 2, 2.
- Chang, C.-C., Hung, Y.-P., & Shih, T. K. (2002). *Future multimedia database and research directions*. Hershey, PA: Idea Group.

Ecklund, D. J., Goebel, V., Plagemann, T., & Ecklund, E. F., Jr. (2002). Dynamic end-to-end QoS management middleware for distributed multimedia systems. *Multimedia Systems*, 8, 431-442.

Elmasri, R., & Navathe, S. (2004). *Fundamentals of database systems* (4th ed.). Boston: Pearson.

Ganz, A., Ganz, Z., & Wongthavarawat, K. (2004). *Multimedia wireless networks: Technologies, standards, and QoS*. Upper Saddle River, NJ: Prentice Hall.

Hillborg, M. (2002). *Wireless XML developer's guide*. Emeryville, CA: McGraw-Hill/Osborne.

Kazantzidis, M. I. (2002). *Adaptive multimedia in wireless IP networks*. Unpublished doctoral dissertation, University of California, Los Angeles.

Miloucheva, I., & Tartarelli, S. (2002). QoS roadmap. In *Next Generation Networks Initiative Consortium*. Retrieved from [www.ist-mome.org/document/qos\\_roadmap\\_final.pdf](http://www.ist-mome.org/document/qos_roadmap_final.pdf)

Ramakrishnan, R., & Gehrke, J. (2003). *Database management systems* (3rd ed.). Boston, : McGraw-Hill.

Shih, T. K. (2001). *An introduction to multimedia database*. Hershey, PA: Idea Group.

Shih, T. K. (2002). *Distributed multimedia databases: Techniques and applications*. Hershey, PA: Idea Group.

Watson, R. (2004). *Data management: Databases and organizations*. New York: Wiley.

## KEY TERMS

**GPRS:** Short for *General Packet Radio Service*, a standard for wireless communications that runs at speeds up to 115 kilobits per second, compared with current GSM's (Global System for Mobile Communications) 9.6 kilobits. GPRS, which supports a wide range of bandwidths, is an efficient use of limited bandwidth and is particularly suited for sending and receiving small bursts of data, such as e-mail and Web browsing, as well as large volumes of data. Retrieved September 20, 2004, from <http://www.webopedia.com/TERM/G/GPRS.html>.

**Mobile Client-Site Multimedia Data Processing Capabilities:** To study and extend the standard QoS mechanism in the context of wireless mobile environments for multimedia applications, the client-site multimedia data processing capabilities can be considered. The related elements such as hardware facilities, user prefer-

ence, mobility coverage, and critical quality acceptance should be included in the extended QoS framework.

**Mobile Multimedia Database Management System (MM-DBMS):** Since a traditional DBMS does not support QoS-based objects, it only concentrates on tables that contain a large number of tuples, each of which is of relatively small size. However, a MM-DBMS should support QoS-sensitive multimedia data types in addition to providing all the facilities for DBMS functions. Once multimedia objects such as images, sound clips, and videos are stored in a database, individual objects of very large size have to be handled efficiently. Furthermore, the crucial point is that mobile MM-DBMSs should have the QoS-based capabilities to efficiently and effectively process the multimedia data in wireless mobile environments.

**Multimedia Database Object Properties:** The properties of a multimedia object refer to the object's QoS-sensitive characteristics. They can be categorized into several attributes. The nature of object can be described by frame size, frame rate, color depth, compression, etc; the quality of object presentation can be determined by delay variation and loss or error rate.

**QoS-Based Multimedia Query Processing and Data Retrieval:** The current multimedia query processing mechanism provides the three procedures in practice. They are (1) search, (2) browsing, and (3) query refinement.

There are four considerations regarding multimedia data retrieval.

- First, the queried objects are successfully presented in the client site under the QoS support.
- Second, since the result queries may contain long audio segments, large images, or long videos, efficient extracting and presenting essential information for clients to browse and select are required.
- Third, the response time that is determined by both the network and database search should be efficiently short.
- Fourth, the query refinements are possible to iterate.

**QoS-Based Query Processing Precision Matrix (QQPPM):** The QQPPM solution covers the specification of client-site QoS preferences and their quantitative relationship to wireless network QoS conditions, and also the specification of the server-site QoS profile and their quantitative relationship to the real-time wireless network QoS conditions. QQPPM will be one of the



functions in MM-DBMSs. The objective of QQPPM is to create a smallest element as a criterion to reduce queried object quality between the server's original data quality and actual quality that the client can access. With the QQPPM, the multimedia query result can be displayed on the client site no matter what the queried object characteristics are and no matter what available network resources have.

**Quality of Service (QoS) Management:** QoS management refers to a set of specific requirements for a particular service provided by a network to users. In general, QoS requirements are in accordance with the perceived QoS based on data transmission and application type. These requirements are usually described by quantitative figures that are more or less related to the

technology behind the network service, and thus a user will find limited flexibility in changing the profile after subscription to the service. In the context of the wireless mobile environment, these requirements can be categorized mainly into four attributed types: (1) *bandwidth*, (2) *timeliness*, (3) *mobility*, and (4) *reliability*.

**UMTS:** Short for *Universal Mobile Telecommunications System*, a 3G mobile technology that will deliver broadband information at speeds up to 2 Mbits/sec. Besides voice and data, UMTS will deliver audio and video to wireless devices anywhere in the world through fixed, wireless and satellite systems. Retrieved September 20, 2004, from [http://www.webopedia.com / TERM /U/UMTS.html](http://www.webopedia.com/TERM/U/UMTS.html). Camera ready copy

# Set Comparison in Relational Query Languages

Mohammad Dadashzadeh  
Oakland University, USA

## INTRODUCTION

Today's de facto database standard, the relational database, was conceived in the late 1960's by Edgar F. Codd at IBM. The relational data model offered the user a logical view of the data that was shielded from consideration of how the data would, in fact, be physically organized in storage. This feat was accomplished in large part by the introduction of relational query languages that would specify the desired set of records in a non-procedural fashion. In contrast to the prevailing record-at-a-time, loop-oriented, procedural query languages of the hierarchical and network database management systems, relational query languages were set-oriented in that they would operate on sets of records (i.e., relations or tables) at-a-time in order to produce the desired set of output records. Codd introduced both a relational algebra and a relational calculus as a basis for dealing with data in relational form. Indeed, he defined what the first relational language was: *Data Sublanguage Alpha* (Codd, 1971).

The non-procedural nature of relational query languages made it possible to envision that end users could be expected to formulate ad hoc queries without resorting to a programmer. To that end, RDBMS adoption was thought to be facilitated by creation of an English-like query language. The language created for this purpose at IBM was called SEQUEL (Structured English Query Language), though it eventually grew in scope to handle other tasks including database modification, definition, authorization, and transaction processing (Chamberlin et al., 1976). At about the same time, another IBM research group produced Query by Example (Zloof, 1975), which because of its graphical interface proved to be easier to use for casual users. However, the wider applicability of SEQUEL led to its adoption and standardization as SQL (Structured Query Language) between 1982 and 1986.

As a relational query language, SEQUEL borrowed features from both relational algebra and relational calculus. However, in an effort to appeal to end users, the expressive power of relational calculus quantification (universal, for all, and existential, there exists) was somewhat sacrificed in favor of algebraic grouping (Group By and SET operations). Unfortunately, the bal-

anced approach of SEQUEL to relational calculus and relational algebra was abandoned in SQL, resulting in undue complexity when formulating queries requiring universal quantification. This article examines the shortcomings of relational query languages in formulating such set comparison queries and proposes solutions to overcome them with minimal effort.

## BACKGROUND

Consider the following relational database about suppliers, parts, and jobs. (The primary key of each relation is underlined.)

SUPPLIER( S#, SName, Status, City )  
PART( P#, PName, Color, City )  
JOB( J#, JName, City )  
SHIPMENT( S#, P#, J#, QTY )

The relation SHIPMENT records the quantity of each part being shipped by each supplier to various jobs. An instance of this database is depicted below.

Table 1. Supplier.db

	<u>S#</u>	SNAME	STATUS	CITY
1	S1	Smith	20.00	London
2	S2	Jones	10.00	Paris
3	S3	Blake	30.00	Paris
4	S4	Clark	20.00	London
5	S5	Adams	30.00	Athens

Table 2. Part.db

	<u>P#</u>	PNAME	COLOR	CITY
1	P1	Nut	Red	London
2	P2	Bolt	Green	Paris
3	P3	Screw	Blue	Rome
4	P4	Screw	Red	London
5	P5	Cam	Blue	Paris
6	P6	Cog	Red	London

## Set Comparison in Relational Query Languages

Table 3. Job.db

	J#	JNAME	CITY
1	J1	Sorter	Paris
2	J2	Punch	Rome
3	J3	Reader	Athens
4	J4	Console	Athens
5	J5	Collator	London
6	J6	Terminal	Oslo
7	J7	Tape	London

Table 4. Shipment.db

	S#	P#	J#	QTY
1	S1	P1	J1	200.00
2	S1	P1	J4	700.00
3	S2	P3	J1	400.00
4	S2	P3	J2	200.00
5	S2	P3	J3	200.00
6	S2	P3	J4	500.00
7	S2	P3	J5	600.00
8	S2	P3	J6	400.00
9	S2	P3	J7	800.00
10	S2	P5	J2	100.00
11	S3	P3	J1	200.00
12	S3	P4	J2	500.00
13	S4	P6	J3	300.00
14	S4	P6	J7	300.00
15	S5	P1	J4	100.00
16	S5	P2	J2	200.00
17	S5	P2	J4	100.00
18	S5	P3	J4	200.00
19	S5	P4	J4	800.00
20	S5	P5	J4	400.00
21	S5	P5	J5	500.00
22	S5	P5	J7	100.00
23	S5	P6	J2	200.00
24	S5	P6	J4	500.00

Now, consider the following queries:

- **Q1:** List the suppliers who ship every red part. (Answer: S5)
- **Q2:** List the suppliers who do not ship to any job located in London. (Answer: S1 and S3)
- **Q3:** List the jobs that are only receiving parts warehoused in London. (Answer: None)
- **Q4:** List the suppliers who are shipping to exactly the same jobs as supplier S1. (Answer: None)
- **Q5:** List the suppliers who are shipping exactly the same parts to jobs located in London as they are shipping to jobs located in Athens. (Answer: S2 and S4)

Each of the above queries involves comparison of sets of values in two tables. For example, in Q1, the set of parts (P# values) associated with each supplier (distinct S# value) in the SHIPMENT table must be examined to determine if it contains the set of parts (P# values) in the PART table sharing the value of “Red” for the COLOR attribute.

Despite their innocuous appearances, queries involving set comparison that tend to arise frequently in online analytical processing (OLAP) situations are especially difficult to formulate in relational query languages (Blanning, 1993; Celko, 1997; Matos & Grasser, 2002; Rao, Badia, & Van Gucht, 1996). This article summarizes the existing approaches and the proposed solutions to set comparison queries in relational algebra, Query by Example, and SQL.

## SET COMPARISON IN RELATIONAL ALGEBRA

In relational algebra (Ramakrishnan & Gehrke, 2003), the DIVISION operator provides the mechanism by which a *restricted* form of set comparison may be directly formulated. To fix ideas, consider the following formulation of query Q1 in relational algebra.

**Q1: List the suppliers who ship every red part.**  
 Temp1 := **SELECTION** (PART) using Color = “Red”  
 Temp2 := **PROJECTION** (SHIPMENT) using S# and P#  
 Result := **DIVISION** (Temp2, Temp1) using Temp2.P# and Temp1.P#

Here, Temp1 is produced by performing the SELECTION operation on the PART table using the selection condition Color = “Red.” Next, the Temp2 intermediate table is formed by projecting columns S# and P# from

the SHIPMENT table. The resulting Temp2 table is divided by Temp1, using P# as the dividing column in each table, to produce the Result table as follows. The rows of Temp2 are grouped according to S# (i.e., all the columns in the table excluding the dividing column). For each group, the set of values for P# (i.e., the dividing column) is compared to see if it *contains* every value in the set of P# (i.e., the dividing column) values in table Temp1. If and only if it does, then the value of S# for that group is placed in the Result table.

Unfortunately, the DIVISION operation only supports *containment* for set comparison. This means that, for example, to *list the suppliers who are shipping to exactly the same jobs as supplier S1* (i.e., query Q4 from above), one must resort to additional operations and intermediate results:

```
Temp1:= SELECTION (SHIPMENT) using S# = "S1"
Temp2:= PROJECTION (Temp1) using J#
Temp3:= SELECTION (SHIPMENT) using S# <> "S1"
Temp4:= PROJECTION (Temp3) using S# and J#
Temp5:= DIVISION (Temp4, Temp1) using Temp4.J#
and Temp2.J#
Temp6:= RESTRICTION (SHIPMENT, Temp2) using
SHIPMENT.J# NOT IN Temp2.J#
Temp7:= PROJECTION (Temp6) using S#
Result:= DIFFERENCE (Temp4, Temp7)
```

Here, the RESTRICTION operation is used to keep only those rows of SHIPMENT for which the J# value does *not* appear in table Temp2. The S# values in such rows represent suppliers who are shipping to at least one job not currently shipped to by S1. Removing such S# values from the result of the DIVISION operation will give us the suppliers who are shipping to *exactly* the same jobs as supplier S1 does.

## A Proposed Solution For Relational Algebra

An improved relational operator called generalized division (GD) overcomes the shortcomings of the DIVISION operation (Dadashzadeh, 1989). The GD operator takes two relations as its inputs (e.g., SHIPMENT and PART) and produces an output table that has the same structure as the first input table (e.g., SHIPMENT). A column from the first (left) relation is designated as the *grouping attribute* (e.g., S# in SHIPMENT), and compatible columns from each relation are specified as the dividing columns (e.g., P# in SHIPMENT and P# in PART). A desired set comparison operation from the following list: **EQUALS**, **IS NOT EQUAL TO**, **CON-**

**TAINS**, **DOES NOT CONTAIN**, **IS IN**, and **IS NOT IN** completes the generalized division operation specification (e.g., **EQUALS**).

Using the GD operation, query Q1 is formulated as follows:

```
Q1: List the suppliers who ship every red part.
Temp1:= SELECTION (PART) using Color = "Red"
Temp2:= GD (SHIPMENT, Temp1)
using (S#) SHIPMENT.P# CONTAINS Temp1.P#
Result:= PROJECTION (Temp2) using S#
```

Here, the GD operator would construct the Temp1 table as follows. First, the rows of SHIPMENT are grouped (sorted) on the basis of their value for the grouping attribute S#. Next, for each group of rows from SHIPMENT (groups are distinguished by the value of the grouping attribute), the set of SHIPMENT.P# values appearing in the rows of that group is determined and compared against the set of Temp1.P# values. If the former set contains the latter, then that group of rows from SHIPMENT is passed to the Temp2 table. Otherwise, they are filtered and would not appear in the output table produced by the GD operation.

The improvement in the expressive power of the GD operation is more clearly seen when the set comparison operation is other than CONTAINS. The following formulations of query Q3 and query Q4 demonstrate this.

```
Q3: List the jobs that are only receiving parts
warehoused in London.
```

```
Temp1:= SELECTION (PART) using City = "London"
Temp2:= GD (SHIPMENT, Temp1)
using (J#) SHIPMENT.P# IS IN Temp1.P#
Result:= PROJECTION (Temp2) using J#
```

```
Q4: List the suppliers who are shipping to exactly
the same jobs as supplier S1.
```

```
Temp1:= SELECTION (SHIPMENT) using S# = "S1"
Temp2:= GD (SHIPMENT, Temp1)
using (S#) SHIPMENT.J# EQUALS Temp1.J#
Result:= PROJECTION (Temp2) using S#
```

An extension of the GD operation called grouped generalized division (GGD) provides for comparison of sets of values associated with matching groups of tuples in two relations (Dadashzadeh, 1989). Such matching of groups of rows in two tables and the associated set comparison are at the heart of the solution to formulating query Q5.

## Set Comparison in Relational Query Languages

**Q5: List the suppliers who are shipping exactly the same parts to jobs located in London as they are shipping to jobs located in Athens.**

Temp1:=**JOIN** (SHIPMENT, JOB) using SHIPMENT.J# = JOB.J#

Temp2:=**SELECTION** (Temp1) using City = "Athens"

Temp3:=**SELECTION** (Temp1) using City = "London"

Temp4:=**GGD** (Temp3, Temp2)

using (Temp3.S#) Temp3.P# **EQUALS** Temp2.P# (Temp2.S#)

Result:=**PROJECTION** (Temp4) using S#

Here, the GGD operator would construct the Temp4 table as follows. First, the rows of Temp3 are grouped on the basis of their value for the grouping attribute Temp3.S#. Similarly, the rows of Temp2 are grouped on the basis of their value for the grouping attribute Temp2.S#. Next, for each group of rows from Temp3 (groups are distinguished by their common value for the grouping attribute), the set of Temp3.P# values appearing in the rows of that group is determined and compared against the set of Temp2.P# values appearing in the rows of the *matching group* (i.e., the group sharing the same value for S#) in Temp2. If the two sets of P# values are equal, then that group of rows from Temp3 is passed to the Temp4 table. Otherwise, they are filtered and would not appear in the output table produced by the GGD operation.

The generality and conciseness offered by the GD and GGD operators present a strong case for their inclusion in a relational algebra interface. The prospect that they can be provided with minimal effort, even in an existing algebraic language, should make their provision difficult to ignore. That prospect is in fact a reality since the GD and GGD operators can be expressed in terms of other relational operations (Dadashzadeh, 1989).

## SET COMPARISON IN QUERY BY EXAMPLE

QBE, like SQL, was developed at IBM, but a number of other DBMSs, such as Paradox, have adopted QBE-like interfaces. Some systems, such as Microsoft Access, offer partial support for graphical queries, which reflects the influence of QBE. In its original definition, QBE provides little support for set comparison queries. In fact, Date (1992) points out that QBE is not relationally complete because it does not manage DIVISION (the algebraic counterpart to universal quantification) appropriately. However, Ramakrishnan and Gehrke (2003) demonstrate that the DIVISION operation can be expressed in IBM's QBE either by using the aggregate operator COUNT (see also Dadashzadeh, 2003; Matos

Figure 1. Suppliers the ship every red part

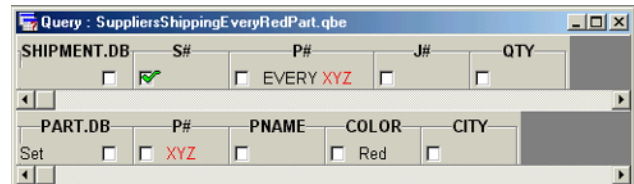


Figure 2. Suppliers that do not ship to any job located in London

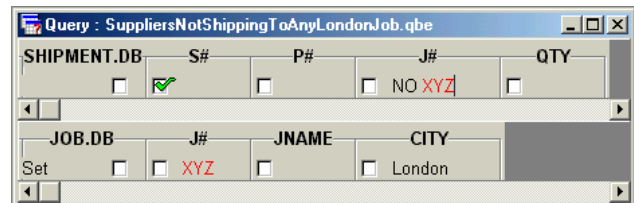


Figure 3. Jobs that only receive parts warehoused in London

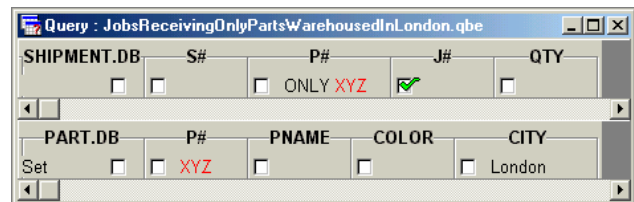
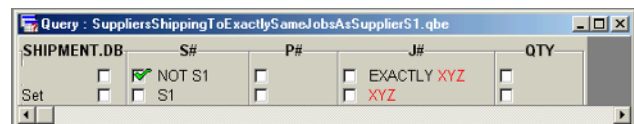


Figure 4. Suppliers shipping to exactly the same jobs as supplier S1



and Grasser, 2002) or by using the data definition commands to create an auxiliary relation or view.

In contrast, Paradox's QBE provides special set operators (SET, EVERY, NO, ONLY, and EXACTLY) that *directly* support the formulation of such queries, as illustrated in Figure 1-4 (Dadashzadeh, 2002):

In this QBE formulation shown in Figure 1, Paradox's SET operator is used to define a set named XYZ as consisting of the P# of all red parts in the PART table. Then, Paradox's set comparison operator EVERY is used to indicate that from the SHIPMENT table only those S#



values should be printed out that appear with EVERY value in the set XYZ.

The clarity afforded by the use of set operators in Paradox's QBE unfortunately falls short when the query calls for comparison of sets of values associated with matching groups of tuples in two relations. Since Paradox considers the use of example elements in its *set* specification for the purpose of identifying a matching group of rows to be ambiguous, queries such as Q5 can only be formulated using a programming-like series of update commands to create and utilize auxiliary tables.

## SET COMPARISON IN SQL

SQL does not provide direct support for comparing two sets. In fact, SQL does not provide operators to perform set intersection or set difference operations where it is required to compare two union-compatible tables for rows that are common to both or that are in one and not in the other. In order to formulate set intersection or set difference operations, the SQL user is expected to construct a query using two of the more difficult concepts in SQL: correlated subquery and the EXISTS function.

To demonstrate, consider the SQL query to list the supplier-part number pairs that reflect a shipment to J1 but that are not involved in shipments to J2. In Oracle's implementation of SQL that supports a set difference operation, this query is formulated as:

```
(SELECT      S#, P#
FROM        SHIPMENT
WHERE       J# = "J1")
MINUS
(SELECT      S#, P#
FROM        SHIPMENT
WHERE       J# = "J2")
```

While, in standard SQL, the formulation becomes:

```
SELECT      S#, P#
FROM        SHIPMENT X
WHERE       J# = "J1"
AND
NOT EXISTS
(SELECT     *
FROM       SHIPMENT
WHERE      J# = "J2"
AND S# = X.S# AND
P# = X.P#)
```

The complexity in formulating set difference and set intersection operations in SQL becomes much more pronounced when dealing with queries involving set comparison for *matching* groups of rows in tables, rather than entire tables, and especially when considering set comparison operations such as equality or containment (Dadashzadeh, 2001). Witness the following formulations of the example queries Q1 and Q5.

**Q1: List the suppliers who ship every red part.**

```
SELECT DISTINCT S#
FROM          SHIPMENT X
WHERE NOT EXISTS
(SELECT      *
FROM        PART
WHERE       Color = "Red"
AND
P# NOT IN
(SELECT     P#
FROM       SHIPMENT
WHERE      S# = X.S#) )
```

**Q5: List the suppliers who are shipping exactly the same parts to jobs located in London as they are shipping to jobs located in Athens.**

```
SELECT      S#
FROM        SUPPLIER X
WHERE       NOT EXISTS
(SELECT     *
FROM       SHIPMENT, JOB
WHERE      S# = X.S#
AND
SHIPMENT.J# = JOB.J#
AND
City = "Athens"
AND
P# NOT IN
(SELECT     P#
FROM       SHIPMENT, JOB
WHERE      S# = X.S#
AND
SHIPMENT.J# = JOB.J#
AND
City = "London" ) )
AND
NOT EXISTS
(SELECT     *
FROM       SHIPMENT, JOB
WHERE      S# = X.S#
AND
SHIPMENT.J# = JOB.J#
AND
City = "London"
AND
```

## Set Comparison in Relational Query Languages

```
P# NOT IN
(SELECT      P#
FROM        SHIPMENT, JOB
WHERE       S# = X.S#
AND
SHIPMENT.J# = JOB.J#
AND
City = "Athens" )
```

### A PROPOSED SOLUTION FOR SQL

The undue complexity in formulating queries involving set comparison was avoided, to a large extent, in SEQUEL2 (Chamberlin et al., 1976), the forerunner of SQL. In SEQUEL2, the EXISTS function is nonexistent. Instead, SEQUEL2 provides explicit support for set comparison in *two* ways. First, the built-in function SET in SEQUEL2 can be used in conjunction with the GROUP BY and HAVING operators to compare a set of values associated with a group of rows with the set of values derived from another table. The set comparison operators supported consist of: IS EQUAL TO; IS NOTEQUAL TO; CONTAINS; DOES NOT CONTAIN; IS IN; and IS NOT IN. The example queries Q1 and Q5 could be formulated in SEQUEL2 as follows.

#### Q1: List the suppliers who ship *every* red part. (SEQUEL2 using SET)

```
SELECT      DISTINCT S#
FROM        SHIPMENT
GROUP BY    S#
HAVING      SET(P#) CONTAINS
            (SELECT      DISTINCT P#
FROM        PART
WHERE       Color = "Red")
```

#### Q5: List the suppliers who are shipping *exactly* the same parts to jobs located in London as they are shipping to jobs located in Athens. (SEQUEL2 using SET)

```
SELECT      DISTINCT S#
FROM        SHIPMENT X, JOB
WHERE       SHIPMENT.J# = JOB.J#
AND
City = "London"

GROUP BY    S#
HAVING      SET(P#)
            IS EQUAL TO
            (SELECT      DISTINCT P#
FROM        SHIPMENT, JOB
WHERE       S# = X.S#
AND
```

```
SHIPMENT.J# = JOB.J#
AND
City = "Athens")
```

The second way in which set comparison can be performed in SEQUEL2 is by direct comparison of compatible sets in the WHERE clause. The example queries Q1 and Q5 could be formulated in SEQUEL2 in the following, decidedly less complex, fashion.

#### Q1: List the suppliers who ship *every* red part. (SEQUEL2 without using SET)

```
SELECT      S#
FROM        SUPPLIER X
WHERE
            (SELECT      DISTINCT P#
FROM        SHIPMENT
WHERE       S# = X.S#)
CONTAINS
            (SELECT      P#
FROM        PART
WHERE       Color = "Red")
```

#### Q5: List the suppliers who are shipping *exactly* the same parts to jobs located in London as they are shipping to jobs located in Athens. (SEQUEL2 without using SET)

```
SELECT      S#
FROM        SUPPLIER X
WHERE       (SELECT      DISTINCT P#
FROM        SHIPMENT, JOB
WHERE       S# = X.S#
AND
SHIPMENT.J# = JOB.J#
AND
City = "London")
IS EQUAL TO
            (SELECT      DISTINCT P#
FROM        SHIPMENT, JOB
WHERE       S# = X.S#
AND
SHIPMENT.J# = JOB.J#
AND
City = "Athens")
```

Clearly, the *only* requirement to support this second approach to formulating set comparisons in SQL is to directly accommodate set comparison operators such as IS EQUAL TO. The other construct employed in the SEQUEL2 formulations above, correlated subqueries, is already in place in SQL. In an unfortunate affront to human factor engineering, the current SQL standard expects the user to reinvent the set comparison opera-

tors using the complex and error-prone EXISTS function (Dadashzadeh, 1992, 1993).

## CONCLUSION

One of the most important promises of the relational data model has been that it frees the decision maker, the manager, from the necessity of resorting to an intermediary, the programmer, in retrieving information from the organization's database in response to unanticipated needs. That promise is founded on the availability of very high-level relational query languages such as SQL, QBE, and relational algebra. Unfortunately, the current implementations of these query languages fail to support users adequately in formulating complex queries involving set comparison that tend to arise in ad hoc decision-making situations. In this article, we have examined these shortcomings and proposed solutions to overcome them with minimal effort.

C. J. Date (2000) has remarked, "A hundred years from now, I'm quite sure, database systems will still be based on Codd's relational foundation." If that would even be partly the case, then there is justifiable hope that the continuing standardization efforts of SQL would finally address the undue complexity in formulating set comparison queries by reintroducing the relational calculus features present in the original definition of SEQUEL.

## REFERENCES

Blanning, R. W. (1993). Relational division in information management. *Decision Support Systems*, 9(4), 313-324.

Celko, J. (1997). *Joe Celko's SQL puzzles & answers*. San Francisco: Morgan Kaufmann.

Chamberlin, D. D., Astrahan, M.M., Eswaran, K.P., Griffiths, P.P., Lorie, R.A., Mehl, J.W., Reisner, P., & Wade, B.W. (1976). SEQUEL2: A unified approach to data definition, manipulation, and control. *IBM Journal of Research & Development*, 20(6), 560-575.

Codd, E. F. (1971). A data base sublanguage founded on the relational calculus. In *Proceedings of the 1971 ACM SIGFIDET Workshop on Data Description, Access and Control* (pp. 35-68). New York: Association for Computing Machinery.

Dadashzadeh, M. (1989). An improved division operator for relational algebra. *Information Systems*, 14(5), 431-437.

Dadashzadeh, M. (1992). A proposed change to the SQL standard. In P. C. Tinnirello (Ed.), *Handbook of systems management: Development and support* (pp. 465-472). Boston: Auerbach.

Dadashzadeh, M. (1993). *A human factor study of set comparison constructs in SQL*. TIMS/ORSA Joint National Meeting.

Dadashzadeh, M. (2001). Set comparison queries in SQL. In S. Becker (Ed.), *Developing quality complex database systems: Practices, techniques, and technologies* (pp. 303-316). Hershey, PA: Idea Group.

Dadashzadeh, M. (2002). Converting Paradox's QBE set queries into Access 2000 SQL. *Review of Business Information Systems*, 6(2), 43-54.

Dadashzadeh, M. (2003). A simpler approach to set comparison queries in SQL. *Journal of Information Systems Education*, 14(4), 345-348.

Date, C. J. (1992). Why quantifier order is important. In C. J. Date & H. Darwen (Eds.), *Relational database writings 1989-1991* (pp. 107-114). Reading, MA: Addison-Wesley.

Date, C. J. (2000). *The database relational model: A retrospective review and analysis*. Reading, MA: Addison-Wesley.

Matos, V. M., & Grasser, R. (2002). A simpler (and better) SQL approach to relational division. *Journal of Information Systems Education*, 13(2), 85-87.

Ramakrishnan, R., & Gehrke, J. (2003). *Database management systems (3rd ed.)*. New York: McGraw-Hill.

Rao, S. G., Badia, A., & Van Gucht, D. (1996). Providing better support for a class of decision support queries. In *Proceedings of the 1996 SIGMOD International Conference on Management of Data* (pp. 217-227). New York: Association for Computing Machinery.

Zloof, M. M. (1975). Query by Example. In *Proceedings of the 1975 National Computer Conference* (pp. 431-438). Montvale, NJ: American Federation of Information Processing Societies.

## KEY TERMS

**Generalized Division:** A generalized version of the binary relational algebra division operation where the tuples of the left table are grouped by an attribute and only those groups whose set of values for another specified attribute satisfies a desired set comparison (e.g.,

## Set Comparison in Relational Query Languages

equality) with a set of similar values from the right table are passed to the output table. The generalized division operator can be expressed in terms of the five principal relational algebra operations.

**QBE:** Query by Example. A graphical query and update language for relational databases introduced by IBM and popularized by Paradox RDBMS.

**RDBMS:** Relational database management system. A software application for managing databases utilizing the relational data model.

**Relational Algebra:** A collection of unary and binary operators that take one or two tables as input and produce a table as output. The relational algebra operators of Cartesian Product, Selection, Projection, set Difference, and Union are considered to be necessary and sufficient for extracting any desired subset of data from a relational database.

**Relational Calculus:** A notation founded on predicate calculus dealing with descriptive expressions that

are equivalent to the operations of relational algebra. Two forms of the relational calculus exist: the tuple calculus and the domain calculus.

**Relational Completeness:** A relational query language is said to be relationally complete if it can express each of the five principal operations of relational algebra.

**Relational Data Model:** A logical way of organizing a database as a collection of interrelated tables. The logical relationship between tables representing related data is accomplished through shared columns, where the primary key column of one table appears as a foreign key column in another.

**Set Comparison Query:** A database query in which the desired records must be found based on a comparison of sets of values using set comparison operations such as inclusion, containment, and equality.

**SQL:** Structured Query Language. The standard language for definition and manipulation of relational databases.

S

# Set Valued Attributes

**Karthikeyan Ramasamy**  
Juniper Networks, USA

**Prasad M. Deshpande**  
IBM Almaden Research Center, USA

## INTRODUCTION

About three decades ago, when Codd (1970) invented the relational database model, it took the database world by storm. The enterprises that adapted it early won a large competitive edge. The past two decades have witnessed tremendous growth of relational database systems, and today the relational model is by far the dominant data model and is the foundation for leading DBMS products, including IBM DB2, Informix, Oracle, Sybase, and Microsoft SQL server. Relational databases have become a multibillion-dollar industry.

However, as these databases grew so did the complexity of the data being stored in them with the emergence of a new class of applications. It quickly became apparent that relational databases suffer from various deficiencies and limitations. Relational database systems support a small, fixed collection of data types (e.g., integers, dates, strings) that has been proven to be adequate for traditional applications. With a new class of applications, more complex data needs to be handled. These complex data include hierarchical data of computer-aided design and modeling (CAD/CAM), multimedia data, and documents. Support for this kind of data requires the database to incorporate abstract data types and type constructors based on object-oriented concepts. This leads to the development of object database systems along two distinct paths:

- **Object-Oriented Database Systems:** These systems were developed with the goal of adding persistence to object-oriented languages that support complex types.
- **Object-Relational Database Systems:** These systems are an attempt to extend the relational database systems with the functionality needed to support complex types.

Object relational systems are characterized by:

- **Abstract Data Types:** They represent the ability to add a new data type into the system that is seamlessly treated as equivalent to built-in types.

The definition of a new data type describes the data fields and the methods that operate on these fields.

- **Type Constructors:** They are used to construct new types by composing base or abstract data types. The major classes of type constructors are composites (records), collections, and references. The class of collections can be further divided into sets, bags, arrays, and lists.
- **Inheritance:** It allows the creation of new data types by derivation from existing types.

This article focuses on the set type constructor popularly referred to as set-valued attributes. Set-valued attributes represent a collection of elements of the same type with uniqueness constraint. To illustrate the usefulness of set-valued attributes, consider the following schema where a product item and the availability of its colors need to be described. In a standard relational schema, we need two tables as follows:

```
PRODUCT(Id: integer, Name: string, Manufacturer:
string)
COLORS(Id: integer, ColorName: string)
```

This schema is complex and similarly the queries will also be more complex (since it requires a join for the relating product and its colors). An instance of these tables is shown in Figure 1.

In an object-relational database system that supports set-valued attributes, we can describe the same by a single table:

```
PRODUCT(Id: integer, Name: string, Manufacturer:
string, Colors: set (string))
```

where the construct **set** indicates that the Colors attribute is a set of strings. This schema is more intuitive and concise from a data modeling and querying perspective. An instance of this table is shown in Figure 2.

This article examines set-valued attributes, their background, and how they are supported in object-relational database systems.



## Set Valued Attributes

Figure 1. An instance of relational schema: *PRODUCT* and *COLORS*

PRODUCT		
<i>Id</i>	<i>Name</i>	<i>Manufacturer</i>
1001	VacuumCleaner	Sears
1002	Food Processor	Kitchen Aid
1003	Juice Blender	Oster

COLORS	
<i>Id</i>	<i>ColorName</i>
1001	Black
1001	Red
1002	White
1003	Black
1004	Blue
1004	Grey

## BACKGROUND

Research in set-valued attributes has been conducted from different perspectives: data modeling, nested relational databases, object-oriented databases, and object-relational databases.

### Data Modeling

Set-valued attributes have long been studied under the context of data modeling, nested relational databases, object-oriented databases, and object-relational databases. Some of the earlier semantic data models (SDM) incorporate sets and collection of entities (Hammer & McLeod, 1981). A further discussion on data modeling using sets can be found in Brodie (1981, 1984). These studies describe how sets can describe the semantic notions of the real world with ease.

### Nested Relational Databases

The nested relational model relaxes the assumption that relational attributes are atomic. In order to extend the power of the relational model, Zaniolo (1983) proposes a query language called GEM. GEM adds sets as a data type. Here the sets are viewed as a logical collection, and the set operations of equality and containment are defined as a part of the query language. It also notes that set operations are very expensive to support in standard relational systems. Extension of relational models with set-valued attributes with reference to statistical databases (SDB) has been studied in detail in Ozsoyoglu, Ozsoyoglu, and Matos (1987). The DASDB projects at the Technical University of Darmstadt (Schek & Scholl, 1989) supported the nested relational algebra with nest and unnest operations. The nest operation is used to break up the set instances into individual tuples duplicating other attributes for each set element. The unnest operation combines multiple tuples on a given attribute to form a set when the rest of the attributes values in the tuple match.

## OBJECT-ORIENTED DATABASES

From the object-oriented database system perspective, there were two different attempts: One adds persistence to object-oriented languages and the other combines the features of a database system with those of an object-oriented language. Such systems supported many collection types, including sets. Three early projects laid the foundation in this area—Gemstone (Copeland & Maier, 1984) was based on Smalltalk, Vbase (Andrews & Harris, 1987) was based on CLU-like language, and Orion (Bannerjee et al., 1987) was based on Common LISP Object System (CLOS). New SQL-like languages were designed to support powerful querying for these systems. These query languages allowed nested queries and universal and existential quantification queries.

### Object-Relational Databases

Object-relational systems typically start from a relational model and its SQL language and build from there. Early systems supported row types and collection types like sets. The best-known research implementations of object-relational database systems are POSTGRES (Stonebraker, 1997; Stonebraker & Kemnitz, 1991) from the University of California, Berkeley and Paradise (Patel et al., 1997) from the University of Wisconsin. POSTGRES supported the dynamic addition of new types, support for complex objects including sets, inheritance, and rules support. Paradise departs from POSTGRES in that it is a parallel object-relational database system. The main contribution is to explore the parallelization of object-relational features in a shared-nothing environment.

## SET-VALUED ATTRIBUTES

In order to incorporate full support for set-valued attributes into object-relational database systems, the following issues must be addressed:

- Extension of data definition language (DDL) and data manipulation language (DML) to accommodate sets
- Storage of tuples/records containing set-valued attributes
- Indexing of set-valued attributes
- Support for efficient set operations

## Extension of DDL and DML

There are many proposals on the extension of data definition language and data manipulation language to support abstract data types and set-valued attributes. GEM (Zaniolo, 1983) extends the relational language QUEL to provide support for the definition and querying of set-valued attributes. It supports the set membership operator *IN* and various other operators, including = (set equality), != (set does not equal), > (superset), >= (superset equal), < (subset), and <= (subset equal). As an illustrative example using our object-relational schema of *PRODUCT*, a query that retrieves the product names having a color of either black or blue can be formulated as:

```
RANGE of P is PRODUCT
RETRIEVE (P.Name)
WHERE "Black" IN P.Colors OR "Blue" IN P.Colors
.....(Q1)
```

$O_2$ , an object-oriented database system, provides a data definition language that allows defining classes (whose instances encapsulate behavior) and types (whose instances are values). Each of the attributes can be either atomic or composite (using set, list, and tuple constructors). It defines an SQL-like syntax for filtering on sets and other composite type instances (Bancilhon, 1992). In the query language of  $O_2$ , query (Q1) can be written as:

```
SELECT P.Name
FROM P IN PRODUCT, C IN P.Colors
WHERE C = "Black" OR C = "Blue".....
.....(Q2)
```

Here the operator *IN* associates a variable referring to individual tuples of the given table. Other proposals include extension of QUEL for POSTGRES (Stonebraker, 1987).

## Storing Sets

In order to efficiently query, the set instances need to be stored in specialized storage representations. Storage representations for sets can be classified based on two orthogonal characteristics: nesting and location. The

characteristic of nesting describes whether the elements in the set are grouped together or scattered. On the other hand, location specifies whether the set elements are stored along with the rest of the attributes in the relation or vertically decomposed and stored separately. Hence, the four feasible representations are nested internal, nested external, unnested internal, and unnested external. Unnested internal is not very useful since it will replicate the rest of the attributes for each set element, leading to update anomalies. These storage representations have been proposed in Stonebraker (1996) and studied in detail from a performance perspective in Ramasamy (2001). Nested internal representation is referred to as a normalized storage model in Hafez and Ozsoyoglu (1988), and a variant of nested external representation is presented as a decomposed storage model (DSM) in Copeland and Khoshafian (1985). Comparison studies in Ramasamy (2001) show that the choice of representations heavily depends on the nature of the queries posed by the user. Unnested representations suffer from the high cost of fetching the tuple from the buffer pool and predicate processing since each set element is stored as a separate tuple. Nested representations perform better with queries containing set predicates since the predicate evaluation takes advantage of clustering of set elements within the tuple.

## Indexing Sets

Lookup queries for sets typically require an indexing structure on the attribute to reduce the response time. The indexing structures for sets can be either nested or unnested. Nested indices treat each set as a single entity, whereas unnested indices treat each set element as an indexable entity. For example, consider the object-relational instance of *PRODUCT*. Let us assume that the record identifiers for each of the rows are *rid1*, *rid2*, and *rid3*, where *rid1* refers to the first row and so on. An unnested index on the attribute *PRODUCT.Colors* will consist of the entries: ("Black," <*rid1*, *rid2*>), ("Blue," <*rid3*>), ("White," <*rid2*>), ("Red," <*rid1*>), and ("Grey," <*rid3*>). The unnested index can be any of the well-known indexing structures like B+-tree, linear hashing, extensible hashing, etc.

Ishikawa, Kitagawa, and Ohbo (1993) examined the use of signature files for subset predicates. The signature file for a set-valued attribute is created by scanning the entire relation and computing the signature for each set instance. The lookup algorithm converts the set being searched into a signature, matches it against all the signatures, and identifies the potential candidates. The problem with the signature files is that there can be "false drops," which require the candidate sets

be actually examined to determine the result. Signature files are not an indexing method as the entire signature file must be scanned when evaluating a lookup. RD-trees (Hellerstein & Pfeffer, 1994) are a variant of the R-tree (Guttman, 1984) that describe the transitive containment relation and allow narrowing the search to the appropriate data tuples unlike signature files. The RD-tree uses signatures as the bounding box equivalent to guide the search from one level of the tree to the next lower level. Similar to R-tree, multiple branches of the tree must be examined to filter the candidates. Inverted files (Brown, Callan, & Croft, 1994) are an example of an unnested index where each set instance is broken into individual elements and an index is created over them that maps each set element to the tuples they occur. They are a popular technique for single element lookups.

### Operations on Sets

Operations on set-valued attributes can be broadly classified as select operations (involves only a single relation) and join operations (involves at least two relations). Select operations retrieve tuples that satisfy a set predicate as illustrated by queries (Q1) and (Q2) in the section on Extension of DDL and DML. The predicate can include equality, subset, superset, and membership. These operations can be executed either by a sequential scan of the entire relation or efficiently by using an index on the set-valued attribute. Join operations involve set-valued attributes on two different relations. Two tuples, one from each relation, are joinable if the join predicate satisfies the set instances from both the tuples. The join predicate can be equality, containment, and intersection. As an example, we can write the query that retrieves the pair of products such that the color availability of one product is the same as the other and more.

```
SELECT P1.Name, P2.Name
FROM PRODUCT P1, PRODUCT P2
WHERE P1.Colors  $\subset$  P2.Colors.....
.....(Q3)
```

Query Q3 involves a self set containment join. Another query that retrieves the pair of products that are available in one or common colors can be written as set intersection as follows:

```
SELECT P1.Name, P2.Name
FROM PRODUCT P1, PRODUCT P2
WHERE P1.Colors  $\cap$  P2.Colors  $\neq \emptyset$ .....
.....(Q4)
```

Set equality algorithms can be evaluated easily using either sort-merge join or hash join. Sort-merge join will require the elements in each set instance be sorted so that a partial order on sets can be defined. Set containment and intersection joins are much more complex to evaluate. Set containment joins have been addressed in Mamoulis (2003), Melnik and Molina (2003), and Ramasamy (2001). These algorithms are based on a divide-and-conquer approach where the tuples of the first relation are partitioned based on the hash value of an element of the set, and the second relation is replicated for each of the elements in the set. Once the partitioning phase is complete, the corresponding partitions are joined together, and the results of each partition are merged for the final result. The algorithms differ in two aspects: (1) the partitioning phase and (2) whether the entire set elements are projected during the intermediate partitioning step or approximation of sets using signatures is projected.

### FUTURE TRENDS

Set-valued attributes and their operations are growing in significance as new applications emerge. These applications can take advantage of them from a performance perspective or a functionality perspective. To quote a few applications, sets can be used to store the customer transaction records into the database so that items that a customer has bought can be clustered as a logical collection to facilitate association rule mining. Self set containment joins have been identified to be useful in the candidate generation step during association rule mining as in Rantzau (2003). In order to store and process XML data, sets have been suggested as an alternative to improve performance (Shanmugasundaram et al., 1999). Still a lot of research effort is needed in the areas of incorporating sets in parallel database systems, more efficient algorithms for join operations in a parallel environment, and estimation of result size of query involving sets.

### CONCLUSION

Set-valued attributes are an important addition to object-relational systems. It is a type constructor that constructs a new type as a collection with a uniqueness constraint. Full support for sets within a database requires extending DDL language to define set type and DML language to pose queries. Storing sets in a database system efficiently requires new types of storage organizations. Query evaluation requires fast lookup of

set-valued attributes using a new type of indices and new class of algorithms for fast evaluation of complex join operations. We have outlined the major issues concerning set-valued attributes and identified the solutions, trade-offs, and their advantages and disadvantages.

## REFERENCES

- Andrews, T., & Harris, C. (1987). Combining language and database advances in an object-oriented development environment. *Proceedings of the ACM Conference on Object Oriented Programming Systems, Languages, and Applications*, October.
- Bancilhon, F., Delobel, C., & Kanellakis, P. (1992). *Building an object-oriented database system: The story of O2*. Morgan Kaufmann.
- Bannerjee, J., Chou, H. T., Garza, J., Kim, W., Woelk, D., & Ballou N. (1987). Data model issues for object-oriented applications. *ACM Transactions on Office Information Systems*, 5(1).
- Brodie, M. L. (1981). Association: Database abstraction. In *Information Modeling and Analysis*. North Holland.
- Brodie, M. L. (1984). On the development of data models. In *Conceptual Modeling*. Springer-Verlag.
- Brown, E. W., Callan, J. P., & Croft, W. B. (1994). Fast incremental indexing for full-text information retrieval. *Proceedings of 20<sup>th</sup> Conference on Very Large Databases*, September.
- Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6), 377-387.
- Copeland, G. P., & Khoshafian, S. (1985). A decomposition storage model. *Proceedings of the ACM SIGMOD Conference on Management of Data*, May.
- Copeland, G. P., & Maier, D. (1984). Making Smalltalk a database system. *Proceedings of the ACM SIGMOD Conference on Management of Data*.
- Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. *Proceedings of ACM SIGMOD International Conference on Management of Data*, June.
- Hafez, A., & Ozsoyoglu, G. (1988). The partial normalized model of nested relations. *Proceedings of International Conference on Very Large Databases*, September.
- Hammer, M., & McLeod, D. (1981). Database description with SDM: A semantic data model. *ACM Transactions on Database Systems*, 6(3).
- Hellerstein, J. M., & Pfeffer, A. (1994). *The RD-tree: An index structure for sets* (Tech. Rep. No. 1252). Madison: University of Wisconsin-Madison, Computer Sciences Department.
- Ishikawa, Y., Kitagawa, H., & Ohbo, N. (1993). Evaluation of signature files as set access facilities in OODBs. *Proceedings of ACM SIGMOD International Conference on Management of Data*.
- Mamoulis, N. (2003). Efficient processing of joins on set-valued attributes. *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*.
- Melnik, S., & Molina, H. G. (2003). Adaptive algorithms for set containment joins. *ACM Transactions on Database Systems*, 28(1).
- Ozsoyoglu, G., Ozsoyoglu, Z. M., & Matos, V. (1987). Extending relational algebra and relational calculus with set-valued attributes and aggregate functions. *ACM Transactions on Database Systems*, 12(4).
- Patel, J., Yu, J., Kabra, N., Tufte, K., Nag, B., Burger, J., et al. (1997). Building a scalable geo-spatial DBMS: Technology, implementation and evaluation. *Proceedings of ACM SIGMOD Conference on Management of Data*.
- Ramasamy, K. (2000). Set containment joins: The good, the bad and the ugly. *Proceedings of the Conference on Very Large Databases*.
- Ramasamy, K. (2001). *Efficient storage and query processing of set-valued attributes*. Unpublished doctoral dissertation, University of Wisconsin-Madison.
- Rantzau, R. (2003). Processing frequent itemset discovery queries by division and set containment join operators. *Proceedings of the Eighth ACM SIGMOD workshop on Research issues in Data Mining and Knowledge Discovery*.
- Schek, H. J., & Scholl, M. H. (1989). The two roles of nested relations in the DASDBS project. In S. Abiteboul, P. C. Fisher, & H. J. Schek (Eds.), *Lecture Notes in Computer Science*. Springer-Verlag.
- Shanmugasundaram, J., Tufte, K., He, G., Zhang, C., DeWitt, D., & Naughton, J. (1999). Relational databases for querying XML documents: Limitations and opportunities. *Proceedings of the Conference on Very Large Databases*.
- Stonebraker, M. (1987). The design of POSTGRES storage system. *Proceedings of the International Conference on Very Large Databases*.

## Set Valued Attributes

Stonebraker, M. (1996). *Object-relational DBMS: The next great wave*. Morgan Kaufmann.

Stonebraker, M., & Kemnitz, G. (1991). The POSTGRES next generation database management system. *Communications of the ACM*, 34(10), 78-92.

Zaniolo, C. (1983). The database language GEM. *Proceedings of the ACM SIGMOD International Conference on Management of Data*.

## KEY TERMS

**Association Rule Mining:** A rule in the form of “if this then that” that associates events in a database; for example, the association between purchased items at a supermarket. The process of examining the data for such rules is called association rule mining.

**Data Definition Language (DDL):** A language used by a database management system which allows users to define the database, specifying data types, structures, and constraints on the data.

**Data Manipulation Language (DML):** A language used by a database management system that allows users to manipulate data (querying, inserting, and updating of data).

**False Drops:** False drops are a property of signature files. Since signature files use hash to activate bits corresponding to the set elements, possibility exists for one or more set elements setting the same bits. When a query signature is evaluated using signatures in signature files, there is a probability that the signatures might match but the actual sets might not match. These are called false drops.

**Set Containment Joins:** A set containment join between relations  $R(a, \{b\})$  and  $S(c, \{d\})$  pairs tuples in relation such that  $\{b\}$  is a subset of  $\{d\}$ .

**Set Equality Joins:** A set equality join between relations  $R(a, \{b\})$  and  $S(c, \{d\})$  pairs tuples in relation such that  $\{b\}$  is equal to  $\{d\}$ .

**Set Intersection Joins:** A set intersection join between relations  $R(a, \{b\})$  and  $S(c, \{d\})$  pairs tuples in relation such that  $\{b\} \cap \{d\} \neq \emptyset$ .

**Signatures:** A signature is a fixed length bit vector that is computed by applying a function  $M$  iteratively to every element  $e$  in the set and setting the bit determined by  $M(e)$ .



# Signature Files and Signature File Construction

**Yangjun Chen**

*University of Winnipeg, Canada*

**Yong Shi**

*University of Manitoba, Canada*

## INTRODUCTION

An important question in information retrieval is how to create a database index which can be searched efficiently for the data one seeks. Today, one or more of the following four techniques have been frequently used: full text searching, B-trees, inversion, and the signature file. Full text searching imposes no space overhead but requires long response time. In contrast, B-trees, inversion, and the signature file work quickly, but need a large intermediary representation structure (index), which provides direct links to relevant data. In this paper, we concentrate on the techniques of signature files and discuss different construction approaches of a signature file.

The signature technique cannot only be used in document databases but also in relational and object-oriented databases. In a document database, a set of semistructured (XML) documents is stored and the queries related to keywords are frequently evaluated. To speed up the evaluation of such queries, we can construct signatures for words and superimpose them to establish signatures for document blocks, which can be used to cut off nonrelevant documents as early as possible when evaluating a query. Especially, such a method can be extended to handle the so-called containment queries, for which not only the key words but also the hierarchical structure of a document has to be considered. We can also handle queries issued to a relational or an object-oriented database using the signature technique by establishing signatures for attribute values, tuples, as well as tables and classes.

## BACKGROUND

The signature file method was originally introduced as a text indexing methodology (Faloutsos, 1985; Faloutsos, Lee, Plaisant & Shneiderman, 1990). Nowadays, however, it is utilized in a wide range of applications, such as in office filing (Christodoulakis, Theodoridou, Ho, Papa, & Pathria, 1986), hypertext systems (Faloutsos et al.), relational and object-oriented databases (Chang & Schek, 1989; Ishikawa, Kitagawa, & Ohbo, 1993; Lee & Lee, 1992; Sacks-Davis, Kent, Ramamohanarao, Thom, & Zobel,

1995; Yong, Lee, & Kim, 1994), as well as data mining (Andre-Joesson & Badal, 1997). It requires much smaller storage space than inverted files and can handle insertion and update operations in databases easily.

A typical query processing with the signature file is as follows: When a query is given, a query signature is formed from the query value. Then each signature in the signature file is examined over the query signature. If a signature in the file matches the query signature, the corresponding data object becomes a candidate that may satisfy the query. Such an object is called a drop. The next step of the query processing is the false drop resolution. Each drop is accessed and examined whether it actually satisfies the query condition. Drops that fail the test are called false drops while the qualified data objects are called actual drops.

A variety of approaches for constructing signature files have been proposed, such as bit-slice files, S-trees, and signature trees. In the following, we overview all of them and discuss a new application of signatures for tree inclusion problem, which is important for containment query evaluation in document databases.

## SIGNATURE FILES AND SIGNATURE FILE ORGANIZATION

### Signature Files

Intuitively, a signature file can be considered as a set of bit strings, which are called signatures. Compared to the inverted index, the signature file is more efficient in handling new insertions and queries on parts of words. But the scheme introduces information loss. More specifically, its output usually involves a number of false drops, which may only be identified by means of a full text scanning on every text block short-listed in the output. Also, for each query processed, the entire signature file needs to be searched (Faloutsos, 1985; Faloutsos, 1992). Consequently, the signature file method involves high processing and I/O cost. This problem is mitigated by partitioning the signature file, as well as by exploiting parallel computer architecture (Ciaccia & Zezula, 1996).

## Signature Files and Signature File Construction

During the creation of a signature file, each word is processed separately by a hashing function. The scheme sets a constant number ( $m$ ) of 1s in the  $[1..F]$  range. The resulting binary pattern is called the word signature. Each text is seen to consist of fixed-size logical blocks and each block involves a constant number ( $D$ ) of noncommon, distinct words. The  $D$  word signatures of a block are superimposed (bit OR-ed) to produce a single  $F$ -bit pattern, which is the block signature stored as an entry in the signature file.

Figure 1 depicts the signature generation and comparison process of a block containing three words (then  $D=3$ ), say “SGML,” “database,” and “information.” Each signature is of length  $F=12$ , in which  $m=4$  bits are set to 1. When a query arrives, the block signatures are scanned and many nonqualifying blocks are discarded. The rest are either checked (so that the “false drops” are discarded; see below) or they are returned to the user as they are. Concretely, a query specifying certain values to be searched for will be transformed into a query signature  $s_q$  in the same way as for word signatures. The query signature is then compared to every block signature in the signature file. Three possible outcomes of the comparison are exemplified in Figure 1: (1) the block matches the query; that is, for every bit set in  $s_q$ , the corresponding bit in the block signature  $s$  is also set (i.e.,  $s \wedge s_q = s_q$ ) and the block contains really the query word; (2) the block doesn’t match the query (i.e.,  $s \wedge s_q \neq s_q$ ); and (3) the signature comparison indicates a match but the block in fact doesn’t match the search criteria (false drop). In order to eliminate false drops, the block must be examined after the block signature signifies a successful match.

In a signature file, a set of signatures is sequentially stored, which is easy to implement and requires low storage space and low update cost. However, when evaluating a query, a full scan of the signature file has to be performed. Therefore, it is generally slow in retrieval.

Figure 1(b) shows a simple signature file. To determine the length of signatures, we use the following formula (Faloutsos, 1985):

$$F \times \ln 2 = m \times D \quad (1)$$

Figure 1. Signature generation, comparison and signature files

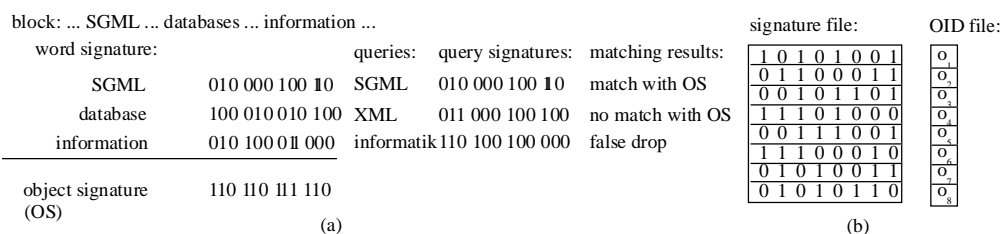
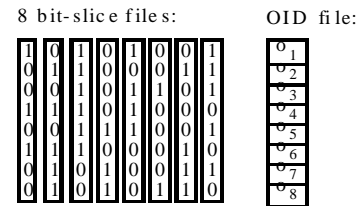


Figure 2. Illustration for bit-slice file



## Bit-Slice Files

A signature file can be stored in a column-wise manner. That is, the signatures in the file are *vertically* stored in a set of files (Ishikawa et al., 1993), i.e., in  $F$  files, in each of which one bit per signature for all the signatures is stored as shown in Figure 2.

With such a data structure, the signatures are checked slice-by-slice (rather than signature-by-signature) to find matching signatures. To demonstrate the retrieval, consider the query signature  $s_q = 10110000$ . First, we check the first bit-slice file and find that only three positions: first, fourth and sixth positions match the first bit in  $s_q$ . Then, we check the second bit-slice file. This time, however, only those three positions will be checked. Since the second bit in  $s_q$  is 0, no positions will be filtered. Next, we check the third bit-slice file against the third bit in  $s_q$ . Because all the three positions are set to 1 in it, the same positions in the next bit-slice file, i.e., in the fourth bit-slice file will be checked against fourth bit in  $s_q$ . Since none of the three positions in the fourth bit-slice file matches this bit, the search stops and reports a *nil*.

From this process, we can see that only part of the  $F$  bit-slice files have to be scanned. So the search cost must be lower than that of a sequential file. However, update cost becomes larger. For example, an insertion of a new set signature requires about  $F$  disk accesses, one for each bit-slice file.

### S-Trees

Similar to a B<sup>+</sup>-tree, an S-tree is a height-balanced multiway tree (Deppisch, 1986). Each internal node corresponds to a page, which contains a set of signatures, and each leaf node contains a set of entries of the form  $\langle s, oid \rangle$ , where the object is accessed by the  $oid$  and  $s$  is its signature. Let  $N$  be the parent node of  $N'$ . Then, there exists a signature in  $N$  whose value is obtained by superimposing all the signatures in  $N'$ . See Figure 3 for illustration.

To retrieve a query signature, we search the S-tree top-down. However, more than one path may be visited. The first signature in the root  $N_1$  leads us to its child node  $N_2$  because the third and fourth bits are set to 1. In  $N_2$ , the second and third signatures match  $s_q$ . Then, we go to the leaf node  $N_4$  and  $N_5$ . In  $N_4$ , we find two matching candidates  $\{o_4, o_5\}$ , and in  $N_5$ , we have only one  $\{o_7\}$ .

The construction of an S-tree is an insertion-splitting process. At the very beginning, the S-tree contains only an empty leaf node, and signatures in a file are inserted into it one by one. When a leaf node  $N$  becomes full, it will be split into two nodes, and at the same time a parent node  $N_{parent}$  will be generated if it does not exist. In addition, two new signatures will put in  $N_{parent}$ . Assume that the capacity of  $N$  is  $K$  (i.e.,  $N$  can accommodate  $K$  signatures.) Then, when we try to insert the  $(K + 1)$ th signature into  $N$ , it has to be split into two nodes  $N_{a\beta}$  and  $N_{b\beta}$ . To do this, we will pick a signature in  $N$  which has the heaviest signature weight (i.e., with the most 1s) in  $N$ . It is called the  $a\beta$ -seed and will be put in  $N_{a\beta}$ . Then, we select a second signature which has the maximum number of 1s in those positions where  $a\beta$  has 0. That is, the signature provides the maximal weight increase to  $a\beta$ . This signature is called the  $b\beta$ -seed and put in  $N_{b\beta}$ . Any of the rest  $K - 1$  signatures are assigned to  $N_{a\beta}$  or  $N_{b\beta}$ , depending on whether it is closer to  $N_{a\beta}$  or  $N_{b\beta}$ . The two new signatures (denoted  $s_{a\beta}$  and  $s_{b\beta}$ ) to be put into the parent node are obtained by superimposing the signatures in  $N_{a\beta}$  and  $N_{b\beta}$ , respectively. See Figure 3(b) for illustration.

The advantage of this method is that the scanning of a whole signature file is replaced by searching several paths in an S-tree. However, the space overhead is almost doubled. Furthermore, due to superimposing, the nodes near the root tend to have heavy weights and thus have low selectivity. This is improved by Tousidou, Bozanis, and Manolopoulos (2002). They elaborate the selection of the  $\alpha$ -seed and the  $\beta$ -seed so that their distance is increased. However, this kind of improvement is achieved at the cost of time, i.e., by checking more signatures, which makes the insertion of a signature into a S-tree extremely inefficient.

In the following, we discuss a quite different method, called signature trees, by means of which all the drawbacks of S-trees are removed.

### Signature Trees

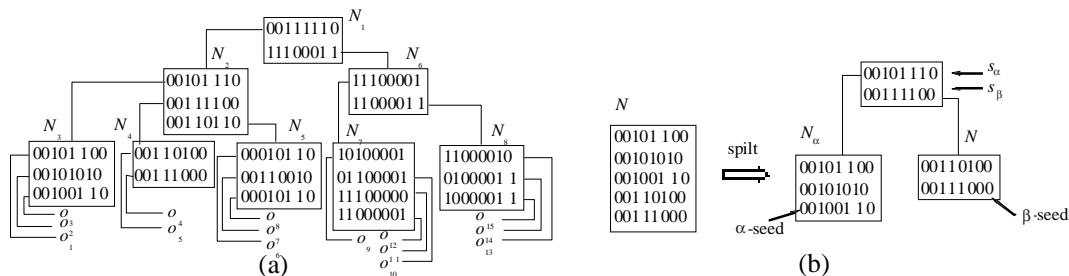
Consider a signature  $s_i$  of length  $F$ . We denote it as  $s_i = s_i[1]s_i[2] \dots s_i[F]$ , where each  $s_i[j] \in \{0, 1\}$  ( $j = 1, \dots, F$ ). We also use  $s_i(j_1, \dots, j_h)$  to denote a sequence of pairs w.r.t.  $s_i$ :  $(j_1, s_i[j_1])(j_2, s_i[j_2]) \dots (j_h, s_i[j_h])$ , where  $1 \leq j_k \leq F$  for  $k \in \{1, \dots, h\}$ .

**Definition 1 (signature identifier).** Let  $S = s_1.s_2 \dots .s_n$  denote a signature file. Consider  $s_i (1 \leq i \leq n)$ . If there exists a sequence:  $j_1, \dots, j_h$  such that for any  $k \neq i$  ( $1 \leq k \leq n$ ) we have  $s_i(j_1, \dots, j_h) \neq s_k(j_1, \dots, j_h)$ , then we say  $s_i(j_1, \dots, j_h)$  identifies the signature  $s_i$  or say  $s_i(j_1, \dots, j_h)$  is an identifier of  $s_i$ .

**Definition 2 (signature tree).** A signature tree for a signature file  $S = s_1.s_2 \dots .s_n$ , where  $s_i \neq s_j$  for  $i \neq j$  and  $|s_k| = F$  for  $k = 1, \dots, n$ , is a binary tree  $T$  such that:

1. For each internal node of  $T$ , the left edge leaving it is always labeled with 0, and the right edge is always labeled with 1.
2.  $T$  has  $n$  leaves labeled 1, 2, ...,  $n$ , used as pointers to  $n$  different positions of  $s_1, s_2 \dots$  and  $s_n$  in  $S$  (signature file). For a leaf node  $u$ ,  $p(u)$  represents the pointer to the corresponding signature in  $S$ .

Figure 3. Illustration for S-tree and node splitting



## Signature Files and Signature File Construction

- Each internal node  $v$  is associated with a number, denoted  $sk(v)$ , which is the bit offset of a given bit position in the block signature pattern. That bit position will be checked when  $v$  is encountered.
- Let  $j_1, \dots, j_h$  be the numbers associated with the nodes on a path from the root to a leaf node labeled  $i$  (then, this leaf node is a pointer to the  $i$ th signature in  $S$ ). Let  $p_1, \dots, p_h$  be the sequence of labels of edges on this path. Then,  $(j_1, p_1) \dots (j_h, p_h)$  makes up a signature identifier for  $s_p, s_i(j_1, \dots, j_h)$ .

**Example 1.** In Figure 4(b), we show a signature tree for the signature file shown in Figure 4(a). In this signature tree, each edge is labeled with 0 or 1, and each leaf node is a pointer to a signature in the signature file. In addition, each internal node is associated with a positive integer (which is used to tell how many bits to skip when searching). Consider the path going through the nodes marked 1, 7, and 4. If this path is searched for locating some signature  $s$ , then three bits of  $s$ :  $s[1]$ ,  $s[7]$ , and  $s[4]$  will have been checked at that moment. If  $s[4] = 1$ , the search will go to the right child of the node marked 4. This child node is marked with 5 and then the fifth bit of  $s$ :  $s[5]$  will be checked.

See the path consisting of the dashed edges in Figure 4(b), which corresponds to the identifier of  $s_6$ :  $s_6(1, 7, 4, 5) = (1, 0)(7, 1)(4, 1)(5, 1)$ . Similarly, the identifier of  $s_3$  is  $s_3(1, 4) = (1, 1)(4, 1)$ ; see the path consisting of the thick edges in Figure 4(b).

Now we discuss how to search a signature tree to model the behavior of a signature file as a filter. Let  $s_q$  be a query signature. The  $i$ -th position of  $s_q$  is denoted as  $s_q(i)$ . During the traversal of a signature tree, the inexact matching is defined as follows:

- Let  $v$  be the node encountered and  $s_q(i)$  be the position to be checked.
- If  $s_q(i) = 1$ , we move to the right child of  $v$ .
- If  $s_q(i) = 0$ , both the right and left child of  $v$  will be visited.

In fact, this definition just corresponds to the signature matching criterion.

**Example 2.** Consider the signature file and the signature tree shown in Figure 4(b) once again.

Assume  $s_q = 00010010000$ . We search the signature tree top-down. First, we check the root  $r$ . Since  $sk(r) = 1$ , we check the first bit in  $s_q$ . It is 0. Then, both the child nodes of  $r$  will be explored. (See the thick edges in Figure 5.) When we visit the left child node  $v$  of  $r$ , the seventh bit in  $s_q$  will be checked since  $sk(v) = 7$ . It is equal to 1. Then, only the right child node of  $v$  will be checked. We repeat this process until all the possible leaf nodes are visited. Obviously, this process is much more efficient than a sequential searching. For this example, only 42 bits are checked (6 bits during the tree search and 36 bits during the signature checking). But by the scanning of the signature file, 96 bits will be checked. In general, if a signature file contains  $N$  signatures, the method discussed above requires only  $O(N/2^l)$  comparisons in the worst case, where  $l$  represents the number of bits set in  $s_q$ , since each bit set in  $s_q$  will prohibit half of a subtree from being visited. Compared to the time complexity of the signature file scanning  $O(N)$ , it is a major benefit.

Due to limitation of space, we don't discuss here the maintenance of signature trees. An interested reader is referred to Chen (2004) for detailed description.

Figure 4. Signature tree

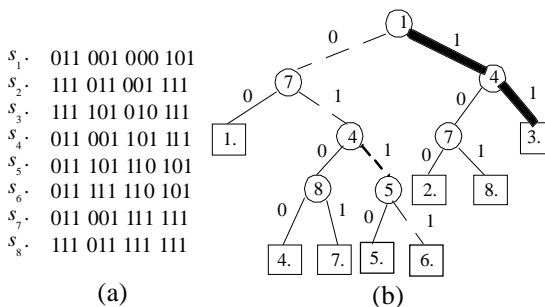
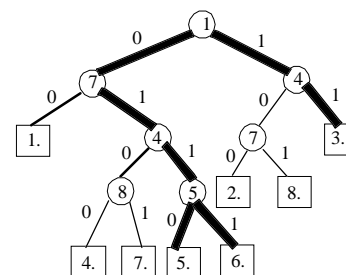


Figure 5. Signature tree search



## INTEGRATING SIGNATURES INTO TREE INCLUSION

In this section, we discuss how to integrate signatures into the tree inclusion problem, which is important to containment queries in document databases. As pointed out in Kilpelainen and Mannila (1995), the evaluation of a containment query is in essence to check whether a query tree is concluded in a document tree.

Let  $T$  be an ordered, rooted tree with root  $v$  and children  $v_1, v_2, \dots, v_i$ . The *postorder* traversal of  $T(v)$  (the tree rooted at  $v$ ) is obtained by visiting  $T(v_k)$ ,  $1 \leq k \leq i$  in order, recursively, and then visiting the  $v$ . The *postorder number*,  $\text{post}(v)$ , of a node  $v \in V(T)$  is the number of nodes preceding  $v$  in the postorder traversal of  $T$ . We define an ordering of the nodes of  $T$  given by  $v \prec v'$  iff  $\text{post}(v) < \text{post}(v')$ . Also,  $v \preceq v'$  iff  $v \prec v'$  or  $v = v'$ . Furthermore, we extend this ordering with two special nodes  $\perp \prec v \prec \top$ . The *left relatives*,  $\text{lr}(v)$ , of a node  $v \in V(T)$  is the set of nodes that are to the left of  $v$  and similarly the *right relatives*,  $\text{rr}(v)$ , are the set of nodes that are to the right of  $v$ .

**Definition 3.** Let  $S$  and  $T$  be rooted labeled trees. We defined an ordered embedding  $(f, S, T)$  as an injective function  $f: V(S) \rightarrow V(T)$  such that for all nodes  $v, u \in V(S)$ ,

1.  $\text{label}(v) = \text{label}(f(v))$ ; (label preservation condition)
2.  $v$  is an ancestor of  $u$  iff  $f(v)$  is an ancestor of  $f(u)$ ; (ancestor condition)
3.  $v$  is to the left of  $u$  iff  $f(v)$  is to the left of  $f(u)$ ; (sibling condition)

Figure 6 shows an example of an ordered inclusion.

In Figure 6(a), we show that the tree on the left can be included in the tree on the right by deleting the nodes labeled:  $d$ ,  $e$ , and  $b$ . Figure 6(b) shows a possible embedding. An embedding is *root preserving* if  $f(\text{root}(S)) = \text{root}(T)$ . Figure 6(b) shows also an example of the root preserving embedding.

A lot of algorithms have been developed to check tree inclusion, such as those reported in Alonso and Schott (1993), Kilpelainen and Mannila (1995), Richter (1997), and Chen (1998). All the methods focus, however, on the

bottom-up strategies to get optimal computational complexities, which are not suitable for the database environment since the algorithms proposed assume that both the target tree (or, say, the document tree) and the pattern tree (or, say, the query tree) can be accommodated completely in main memory. Recently, a top-down algorithm was proposed (Chen & Chen, 2004) which has the same time complexity as the best bottom-up algorithm but needs no extra space. More importantly, it works well in a database environment for the reason that it checks a target tree in a top-down fashion, and each time only part of the tree is manipulated. Furthermore, it can be combined with *signatures* to speed up query evaluation.

The top-down algorithm can be outlined as follows. A detailed description can be found in Chen and Chen (2004).

**Algorithm** *top-down-tree-inclusion*

1. Let  $r_1$  and  $r_2$  be the roots of  $T$  and  $S$ , respectively.
2. Let  $T_1, \dots, T_n$  be the subtrees of  $r_1$ , and  $S_1, \dots, S_m$  be the subtrees of  $r_2$ .
3. If  $r_1$  doesn't match  $r_2$ , try to find an  $i$  ( $1 \leq i \leq n$ ) such that  $T_i$  includes the whole  $S$ .
4. If  $r_1$  matches  $r_2$ , try to find  $i_1, \dots, i_k$  such that  $T_{i_1}$  contains  $S_1, \dots, S_{j_1}$ ,  $T_{i_2}$  contains  $S_{j_1+1}, \dots, S_{j_2}$ , and  $T_{i_k}$  contains  $S_{j_{k-1}+1}, \dots, S_{j_k}$ , where  $S_{j_k} = S_m$ .

To speed up this process, we assign each label a signature, and construct the signatures for non-leaf nodes as follows.

**Definition 4.** Let  $v$  be a node in a tree  $T$ . If  $v$  is a leaf node, its signature  $s_v$ . If  $v$  is a non-leaf node, let  $v_1, \dots, v_n$  be its children, then  $s_v = s \vee s_{v_1} \vee \dots \vee s_{v_n}$ , where  $s$  represents the signature for the label associated with  $v$ , and  $s_{v_1}, \dots, s_{v_n}$  are the signatures of  $v_1, \dots, v_n$ , respectively.

**Example 3.** Consider the tree shown in Figure 7(a). If the signatures assigned to the labels are those shown in Figure 7(b), each node in the tree will have a signature as shown in Figure 7(c).

Figure 6. Illustration of tree inclusion

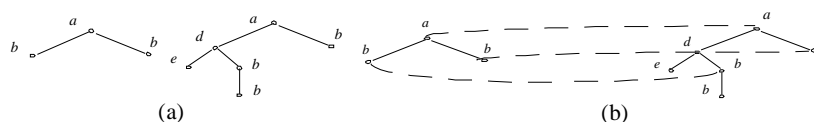
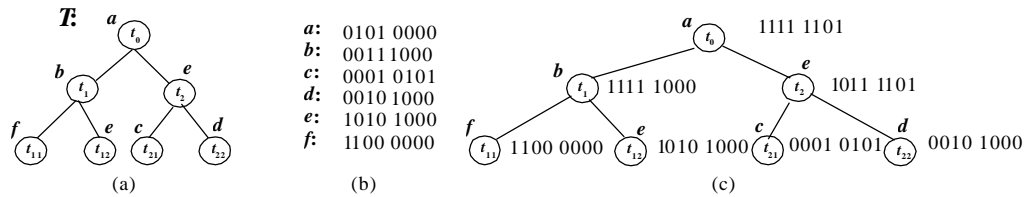




Figure 7. Node signatures



Then, each time we check a node  $u$  in  $S$  against a node  $v$  in  $T$ , we will first check their signatures. If they don't match, the subtree rooted at  $v$  will be cut off and not be searched any more, reducing the time overhead greatly.

Here, an important problem is how to determine the length of signatures. Due to the superimposing of signatures along the tree paths, Equation (1) (shown in the section Signature Files) is not useful any more since it was established only for the simple structure of sequential signature files. However, if the length of signatures is not properly determined, as in an S-tree, the signatures near the root will be very heavy and the selectivity will be reduced dramatically. For this reason, we make the following analysis and develop a new way to estimate the signature length in such a way that the above problem can be removed.

Consider two signatures  $s_1$  and  $s_2$ . Assume that both of them are of length  $F$  and with  $m_1$  and  $m_2$  bits set to 1, respectively. Now, let  $s = s_1 \vee s_2$ . Obviously,  $s$  will possibly contain more 1s. To keep the ratio of 1s in  $s$  not increased,  $s$  should be set longer. The question is: How long should  $s$  be? Let  $l$  be the number of 1s in  $s$  and denote  $\delta = l - m'$ , where  $m' = \max(m_1, m_2)$ . Then,  $F + c\delta$  should be a reasonable length for  $s$ , where  $c$  is a constant and should be tuned for different applications. The value of  $d$  can be estimated as follows.

Let  $\lambda$  be a random variable representing the number of positions, in which both  $s_1$  and  $s_2$  have 1s. Then, the mathematical expectation of  $\lambda$  can be calculated as below:

$$E\lambda = 1 \cdot p(\lambda = 1) + 2 \cdot p(\lambda = 2) + \dots + m'' \cdot p(\lambda = m'') \quad (2)$$

where  $m'' = \min(m_1, m_2)$  and  $p(\lambda = i)$  represents the probability of  $\lambda$  equal to  $i$ . To calculate this probability, we use the following formula:

$$p(\lambda = i) = \frac{\binom{F - m_2}{m_1 - i} \binom{m_2}{i}}{\binom{F}{m_1}} \quad (3)$$

Note that  $l = m_1 + m_2 - \lambda$ . Then, we have  $\delta = l - m = m_1 + m_2 - \lambda - \max(m_1, m_2)$ .

Using the above formulas, we can determine the length of signatures for a tree as follows. First, we calculate the average number of key words in all the leaf nodes, which is used as the value of  $D$  to determine the initial values of  $F$  and  $m$  using Equation (1). Then, we compute the lengths of signatures for the internal nodes in a bottom-up way. That is, we first calculate the lengths for all those nodes, each of which is a parent of some leaf nodes. Then, we compute the lengths for the nodes at a higher level. This process is repeated until the length of the signature for the root is computed, which will be used as the length of all the signatures to be generated.

## FUTURE TREND

As discussed above, the signature file is a useful technique for text indexing and query evaluation in databases. It can also be utilized for some problems in the computational graph theory, i.e., for the tree inclusion problem. As future research, we will concentrate on an interesting issue: how to reduce the size of a signature file. This may be done by elaborating Equation (1), shown in the section Signature Files, which is only an empirical formula. What we want is to find a mathematical method to determine, for a given set of key words which are distributed in a collection of blocks, the minimum length of the signatures and the best choice of the number of bits that are set to 1.

## CONCLUSION

In this article, four methods for constructing signature files are described. They are the sequential signature file, the bit-slice signature file, the S-tree, and the signature tree. Among these methods, the signature file has the simplest structure and is easy to maintain, but it is slow for information retrieval. In contrast, the bit-sliced file and the S-tree are efficient for searching but need more time for maintenance. In addition, an S-tree needs much more

space than a sequential signature file or a bit-slice file. The last method, i.e., the signature tree structure, improves the S-tree by using less space for storage and less time for searching. Finally, as an important application, the signatures can be integrated into the top-down tree inclusion strategy to speed up the evaluation of containment queries. This can also be considered as a quite different way to organize a signature file.

## REFERENCES

- Alonso, L., & Schott, R. (1993). On the tree inclusion problem. *Proceedings of Mathematical Foundations of Computer Science* (pp. 211-221).
- Andre-Joesson, H., & Badal, D. (1997). Using signature files for querying time-series data. *Proceedings of the 1st European Symposium on Principles of Data Mining and Knowledge Discovery*.
- Chang, W. W., & Schek, H. J. (1989). A signature access method for the STARBURST database system. *Proceedings of the 19th VLDB Conference* (pp. 145-153).
- Chen, W. (1998). More efficient algorithm for ordered tree inclusion. *Journal of Algorithms*, 26, 370-385.
- Chen, Y. (2004). Building signature trees into OODBs. *Journal of Information Science and Engineering*, 20, 275-304.
- Chen, Y., & Chen, Y. B. (2004). An efficient top-down algorithm for tree inclusion. *Proceedings of the 18th International Conference Symposium on High Performance Computing System and Application*.
- Christodoulakis, S., Theodoridou, M., Ho, F., Papa, M., & Pathria, A. (1986). Multimedia document presentation, information extraction and document formation in MINOS—A model and a system. *ACM Transactions On Office Information Systems*, 4(4), 345-386.
- Ciaccia, P., & Zezula, P. (1996). Declustering of key-based partitioned signature files. *ACM Transactions on Database Systems*, 21(3), 295-338.
- Deppisch, U. (1986). S-tree: A dynamic balanced signature index for office retrieval. *ACM SIGIR Conference*, 77-87.
- Faloutsos, C. (1985). Access methods for text. *ACM Computing Surveys*, 17(1), 49-74.
- Faloutsos, C. (1992). Signature files. In W. B. Frakes & R. Baeza-Yates (Eds.), *Information retrieval: Data structures & algorithms* (pp. 44-65). NJ: Prentice Hall.
- Faloutsos, C., Lee, R., Plaisant, C., & Shneiderman, B. (1990). Incorporating string search in hypertext system: User interface and signature file design issues. *HyperMedia*, 2(3), 183-200.
- Ishikawa, Y., Kitagawa, H., & Ohbo, N. (1993). Evaluation of signature files as set access facilities in OODBs. *Proceedings of ACM SIGMOD International Conference on Management of Data* (pp. 247-256).
- Kilpelainen, P., & Mannila, H. (1995). Ordered and unordered tree inclusion. *SIAM Journal of Computing*, 24, 340-356.
- Lee, W., & Lee, D. L. (1992). Signature file methods for indexing object-oriented database systems. *Proceedings of ICIC'92—2nd International Conference on Data and Knowledge Engineering: Theory and Application* (pp. 616-622).
- Richter, T. (1997). A new algorithm for the ordered tree inclusion problem. In *Lecture Notes of Computer Science: Vol. 1264. Proceedings of the 8th Annual Symposium on Combinatorial Pattern Matching, CPM* (pp. 150-166).
- Sacks-Davis, R., Kent, A., Ramamohanarao, K., Thom, J., & Zobel, J. (1995). Atlas: A nested relational database system for text application. *IEEE Transactions on Knowledge and Data Engineering*, 7(3), 454-470.
- Tousidou, E., Bozaris, P., & Manolopoulos, Y. (2002). Signature-based structures for objects with set-values attributes. *Information Systems*, 27(2), 93-121.
- Yong, H. S., Lee, S., & Kim, H. J. (1994). Applying signatures for forward traversal query processing in object-oriented databases. *Proceedings of 10th International Conference on Data Engineering* (pp. 518-525).

## KEY TERMS

**Bit-Slice Signature File:** A bit-slice file is a file in which one bit per signature for all the signatures is stored. For a set of signatures of length  $F$ ,  $F$  bit-slice files will be generated.

**S-tree:** An S-tree is a height-balanced multiway tree. Each internal node corresponds to a page, which contains a set of signatures, and each leaf node contains a set of entries of the form  $\langle s, oid \rangle$ , where the object is accessed by the  $oid$  and  $s$  is its signature.

**Sequential Signature File:** A signature file is a set of signatures stored in a file in a sequential way.

## ***Signature Files and Signature File Construction***

**Signature Identifier:** A signature identifier for a signature in a signature file is a positioned bit string which can be used to identify it from others.

**Signature Tree:** A signature tree is an index structure in which each path represents a signature identifier for the signature pointed to by the corresponding leaf node.

**Signatures:** A bit string generated for a key word by using a hash function.

**Tree Inclusion:** Let  $T$  and  $S$  be ordered, labeled trees.  $S$  is said to be included in  $T$  if there is a sequence of delete operations performed on  $T$  which make  $T$  isomorphic to  $S$ .

**S**

# Similarity Search in Time Series Databases

**Maria Kontaki**

*Aristotle University of Thessaloniki, Greece*

**Apostolos N. Papadopoulos**

*Aristotle University of Thessaloniki, Greece*

**Yannis Manolopoulos**

*Aristotle University of Thessaloniki, Greece*

## INTRODUCTION

In many application domains, data can be represented as a series of values (*time series*). Examples include stocks, seismic signals, audio, and many more. Similarity search in time series databases is an important research direction. Several methods have been proposed in order to provide algorithms for efficient query processing in the case of static time series of fixed length. Research in this field has focused on the development of effective transformation techniques, the application of dimensionality reduction methods, and the design of efficient indexing schemes. These tools enable the process of *similarity queries* in time series databases. In the case where time series are continuously updated with new values (*streaming time series*), the similarity problem becomes even more difficult to solve, since we must take into consideration the new values of the series. The dynamic nature of streaming time series makes the methods proposed for the static case inappropriate. To attack the problem, significant research has been performed towards the development of effective and efficient methods for streaming time series processing. In this paper, we introduce the most important issues concerning similarity search in static and streaming time series databases, presenting fundamental concepts and techniques that have been proposed by the research community.

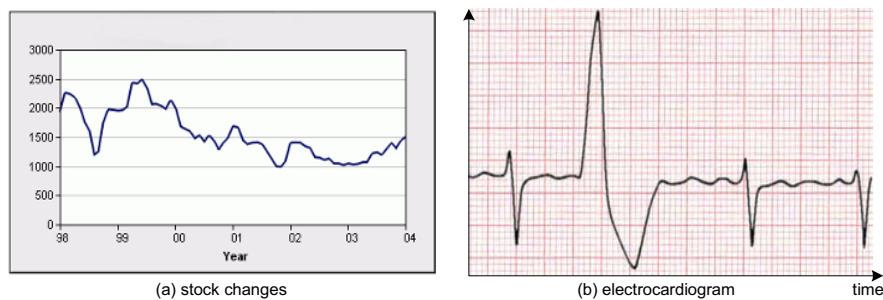
## BACKGROUND

Time series are used in a broad range of applications, modeling data that change over time. For example, stock changes, audio signals, seismic signals, and electrocardiograms, can be represented as time series data. In fact, any measurement that changes over time can be represented as a time series. Two simple time series examples are depicted in Figure 1.

We differentiate between two types of time series, namely: (1) *static time series* and (2) *streaming time series*. In the first case, we assume that the time series is composed of a finite number of sample values, whereas in the second case, the size of the series is increasing since new values are appended. For example, if the data correspond to stock prices for the year 2004, then we can use static time series to capture the stock prices of the time period of interest. On the other hand, if there is a need for continuous stock monitoring as time progresses, then streaming time series are more appropriate.

Streaming time series is a special case of streaming data, which nowadays are considered very important, and there is an increasing research interest in the area. Traditional database methods cannot be applied directly to data streams. Therefore, new techniques and algorithms are required in order to guarantee efficient and effective query processing in terms of the CPU time and the number

Figure 1. Examples of time series data



of disk accesses. The most important difficulty that these techniques must address is the continuous change, which poses serious restrictions.

The purpose of a time series database is to organize the time series in such a way that user queries can be answered efficiently. Although user queries may vary according to the application, there are some fundamental query types that are supported:

- *whole-match queries*, where all time series have the same length; and
- *subsequence-match queries*, where the user's time series is smaller than the time series in the database, and therefore, we are interested in time series which contain the user's time series.

In contrast to traditional database systems, time series databases may contain erroneous or noisy data. This means that the probability that two time series have exactly the same values in the same time instances is very small. In such a case, *exact search* is not very useful, and therefore, *similarity search* is more appropriate. There are three basic types of similarity queries:

- **Similarity range query:** given a user time series  $Q$  and a distance  $e$ , this query retrieves all time series that are within distance  $e$  from  $Q$ .
- **Similarity nearest-neighbor query:** given a user time series  $Q$  and an integer  $k$ , this query retrieves the  $k$  series that are closer to  $Q$ .
- **Similarity join query:** given two sets of time series  $U, V$  and a distance  $e$ , this query retrieves all pairs  $(u, v)$   $u \in U$  and  $v \in V$  such that the distance between  $u$  and  $v$  is less or equal to  $e$ .

It is evident from the above definitions that in order to express similarity between two time series objects, a distance measure  $D$  is required. This distance measure

usually ranges between 0 and 1. If two time series  $u$  and  $v$  are similar, then the value  $D(u, v)$  should be close to 1, whereas if they are dissimilar, then  $D(u, v)$  should be close to 0. Similarity search can be applied for whole-match queries and subsequence-match queries as well, for static or streaming time series.

## SIMILARITY SEARCH IN TIME SERIES

We begin our study with methods proposed for static time series. Streaming time series are considered later in this section. The efficient processing of similarity queries requires the addressing of the following important issues:

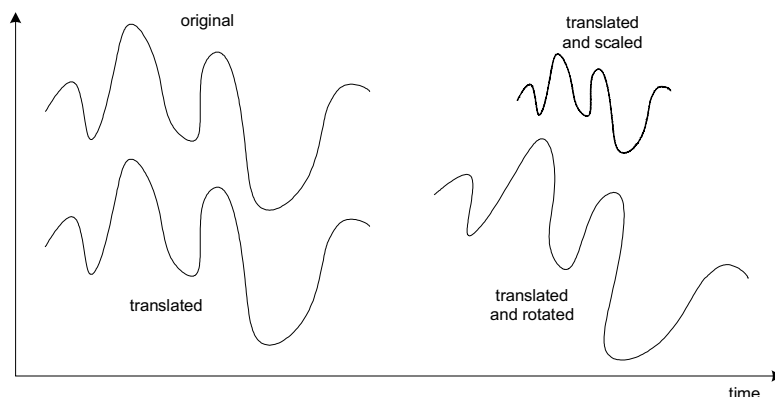
- the definition of a meaningful distance measure  $D$  in order to express the similarity between two time series objects,
- the efficient representation of time series data, and
- the application of an appropriate indexing scheme in order to quickly discard database objects that cannot contribute to the final answer.

Assuming that each time series has a length of  $m$ , then it is natural to think that each time series is represented as a vector in the  $m$ -dimensional space. In such a case, the similarity between two time series  $u$  and  $v$  can be expressed as the Euclidean distance:

$$D(u, v) = \sqrt{\sum_{i=1}^m (u[i] - v[i])^2}$$

where  $u[i]$ ,  $v[i]$  is the value of  $u$  and  $v$  for the  $i$ -th time instance. The Euclidean distance has been widely used as a similarity measure in time series literature (Agrawal,

Figure 2. Examples of translation, rotation and scaling of time series





Faloutsos & Swami, 1993; Chan & Fu, 1999; Faloutsos, Ranganathan & Manolopoulos, 1994; Kontaki & Papadopoulos, 2004) because of its simplicity.

Several alternative distance functions have been proposed in order to allow translation, rotation, and scaling invariance. Consider for example the time series depicted in Figure 2. Note that although all time series have the same shape, they will be considered different if the Euclidean distance is used to express similarity. Translation, rotation, and scaling invariance is studied in Agrawal et al. (1995); Chan and Fu (1999); Yi, Jagadish, and Faloutsos (1998); and Yi and Faloutsos (2000).

The main shortcoming of the Euclidean distance is that all time series must have equal length, which is a significant restriction. If time series are sampled using different time intervals, then their length will not be the same, and therefore, the Euclidean distance cannot be applied. In order to express similarity between time series of different lengths, other more sophisticated distance measures have been proposed (Park et al., 2000; Yi et al., 1998). One such distance measure is *Time Warping* (TW) that allows time series to be stretched along the time axis. The time warping distance maps each element of a time series  $u$  to one or more elements of another time series  $v$ . Given two time series  $u$  and  $v$ , the time warping distance  $D_{TW}(u, v)$  is defined as follows (several variations have been proposed):

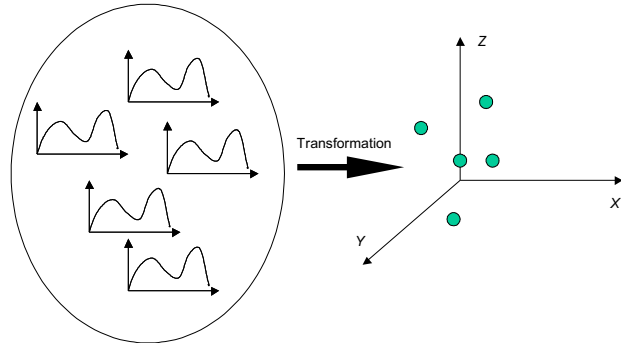
$$D_{TW}(u, v) = |u[1] - v[1]| + \min \begin{cases} D_{TW}(u, v[2: *]) \\ D_{TW}(u[2: *], v) \\ D_{TW}(u[2: *], v[2: *]) \end{cases}$$

where  $u[2: *]$  and  $v[2: *]$  denote the suffix of  $u$  and  $v$  respectively. The TW distance has been used extensively in pattern matching for voice, audio, and other types of signals (e.g., electrocardiograms). The computation cost of the TW is much higher than that of the Euclidean distance.

Many string similarity functions as the Edit Distance (Bozkaya, Yazdani & Ozsoyoglu, 1997) and the Longest Common Subsequence have been modified in order to match similar time series. These similarity functions can be used in time series with different lengths or different sampling rates because some elements may be unmatched.

The number of samples of each time series may range from a few to hundreds or thousands. Therefore, representing each time series as an  $m$ -dimensional vector may result in performance degradation during query processing, due to high computation costs of the distance function. It is preferable to compute the distance as efficiently as possible towards increased processing performance. To attack this problem, *dimensionality reduction* is applied to the time series in order to transform them to a more

Figure 3. Applying dimensionality reduction using transformation



manageable representation. Figure 3 illustrates an example.

One of the first dimensionality reduction techniques applied to time series is the *Discrete Fourier Transform* (DFT), which transforms a time series to the frequency domain. For many real time series (e.g., stock changes), it is observed that most of the information is concentrated in the first few DFT coefficients. Therefore, each time series can be represented by using only a few real numbers, instead of using all the values in the time domain. DFT has been successfully used for whole-match and subsequence-match in static or streaming time series (Agrawal et al., 1993; Faloutsos et al., 1994; Kontaki & Papadopoulos, 2004; Yi & Faloutsos, 2000). Among other dimensionality reduction techniques, we note the *Singular Value Decomposition* (SVD), the *Discrete Wavelet Transform* (DWT), and *FastMap* which have been successfully applied in time series databases.

Note that dimensionality reduction is a lossy operation, since after the transformation of the time series some information is lost. This means that the transformed data is an approximation of the original data, and the latter must be retained in order to be available for reference. The original and the transformed data are used for query processing by means of the *filter-refinement* processing technique:

- During the *filter step*, the approximations (transformed data) are investigated in order to quickly discard time series that cannot contribute to the final answer. The result of this step is a set of *candidate* time series that may be part of the final answer.
- Candidates are investigated further in the *refinement step*, by accessing the original time series in the time domain. Candidate objects that do not satisfy query constraints are characterized as *false alarms* and are discarded.

It is important that the filter step be very efficient with respect to the processing performance and the number of candidates determined. The number of candidates depends on the transformation technique used to produce the approximations, whereas the processing performance depends heavily on the *indexing scheme* applied. If the time series approximations are vectors in a multi-dimensional space, then *spatial access methods* can be used to organize data hierarchically. One of the most influential spatial access methods is the R-tree (Guttman, 1984) and the R\*-tree (Beckmann et al., 1990), which is one of its successful variations. The multi-dimensional approximations are organized in an efficient way in order to speed up query processing. The R\*-tree manages to discard quickly a large portion of the database, and therefore helps significantly in the efficient processing of the filter step. Efficient algorithms have been proposed for similarity range search, similarity nearest-neighbor search, and similarity join. An R\*-tree example for a two-dimensional point dataset is illustrated in Figure 4.

If the dimensionality of the transformed data is still large after the application of the dimensionality reduction method, then indexing schemes for high-dimensional data can be used. These schemes are influenced by the R\*-tree access method, and they use several optimization techniques in order to attack the dimensionality curse problem. Among these sophisticated access methods, we highlight the TV-tree (Lin, Jagadish & Faloutsos, 1995) and the X-tree (Berchtold, Keim & Kriegel, 1996).

Although a significant amount of research work has been performed for static time series, the field of streaming time series is quite immature, since the data stream model has been recently introduced (Babcock et al. 2002; Babu & Widom, 2001; Gilbert et al., 2003). The difficulty in a streaming time series database is that new values for the time series continuously arrive. This characteristic yields existing techniques for static time series inefficient be-

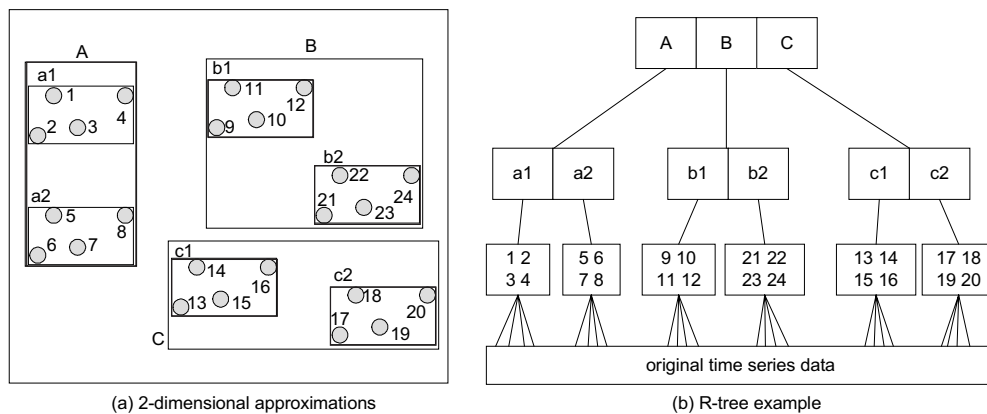
cause the index must be updated continuously for every new value.

Similarity queries in streaming time series have been studied in Gao and Wang (2002a) where whole-match queries are investigated by using the Euclidean distance as the similarity measure. A prediction-based approach is used for query processing. The distances between the query and each data stream are calculated using the predicted values. When the actual values of the query are available, the upper and lower bound of the prediction error are calculated, and the candidate set is formed using the predicted distances. Then, false alarms are discarded. The same authors have proposed two different approaches, based on prefetching (Gao & Wang, 2002b; Gao, Yao & Wang, 2002c).

The aforementioned research efforts examine the case of whole-match queries, where the data are static time series, and the query is a streaming time series. In Liu and Ferhatosmanoglu (2003), the authors present a method for query processing in streaming time series where both the query object and the data are streaming time series. The VA-stream and VA+-stream access methods have been proposed, which are variations of the VA-file (Weber, Schek & Blott, 1998). These structures are able to generate a summarization of the data and enable the incremental update of the structure every time a new value arrives. The performance of this approach is highly dependent on the number of bits associated with each dimension.

In Kontaki and Papadopoulos (2004) a different approach is followed in order to provide a flexible technique for similarity range queries where both data and queries are streaming time series. The proposed method (IDC-Index) is based on the R\*-tree which is used as the indexing scheme for the underlying time series approximations. The dimensionality reduction technique applied to the original time series is based on an incremental compu-

Figure 4. An R\*-tree example for 2-dimensional approximations



tation of the DFT which avoids recomputation. Moreover, the R-tree is equipped by a deferred update policy in order to avoid index adjustments every time a new value for a streaming time series is available. Experiments performed on synthetic random walk time series and on real time series data have shown that the proposed approach is very efficient in comparison to previously proposed methods.

## FUTURE TRENDS

The research interest in the last years has focused on the streaming time series. Apart from the investigation of more efficient techniques for similarity search, there is significant work performed towards *data mining* of streaming data. The challenge is to overcome the difficulty of continuous data change and apply *clustering* algorithms to streaming time series. Some interesting results have been reported (Guha et al., 2003).

Another important research direction is the management of *continuous queries* over streaming time series. In this case, users pose queries that must be continuously evaluated for a time interval. Therefore, when a new value for a time series arrives, the value must be used to determine which queries are satisfied. Continuous queries over data streams are studied in Babcock et al. (2002), Chandrasekaran and Franklin (2002), Gao and Wang (2002a, 2002b) and Gao et al. (2002c).

So far we have focused on one-dimensional time series, since there is only one measurement that changes over time. However, there are applications that require the manipulation of multi-dimensional time series. For example, consider an object that changes location, and we are interested in tracking its position. Assuming that the object moves in the two-dimensional space, there are two values ( $x$  and  $y$  coordinates) that change over time. Some interesting research proposals for multi-dimensional time series can be found (Vlachos et al., 2003).

## CONCLUSION

Time series data are used to model values that change over time, and they are successfully applied to diverse fields such as online stock analysis, computer network monitoring, network traffic management, and seismic wave analysis. In order to manipulate time series effectively and efficiently, sophisticated processing tools are required from the database viewpoint.

Time series are categorized as static or streaming. In static time series, each sequence has a static length, whereas in the streaming case, new values are continu-

ously appended. This dynamic characteristic of streaming time series poses significant difficulties and challenges in storage and query processing.

A fundamental operation in a time series database system is the processing of similarity queries. To achieve this goal, there is a need for a distance function, an appropriate representation, and an efficient indexing scheme. By using the filter-refinement processing technique, similarity range queries, similarity nearest-neighbor queries, and similarity join queries can be answered very efficiently.

## REFERENCES

- Agrawal, R., Faloutsos, C., & Swami, A. (1993). Efficient similarity search in sequence databases. *Proceedings of FODO* (pp. 69-84), Evanston, Illinois.
- Agrawal, R., Lin, K.-I., Sawhney, H.S., & Swim, K. (1995). Fast similarity search in the presence of noise, scaling, and translation in time-series databases. *Proceedings of VLDB*, Zurich, Switzerland.
- Babcock, B., Babu, S., Datar, M., Motwani, R., & Widom, J. (2002). Models and issues in data stream systems. *Proceedings of ACM PODS* (pp. 1-16), Madison, Wisconsin.
- Babu, S., & Widom, J. (2001). Continuous queries over data streams. *SIGMOD Record*, 30(3), 109-120.
- Beckmann, N., Kriegel, H.-P., Schneider, R., & Seeger, B. (1990). The R\*-tree: An efficient and robust access method for points and rectangles. *Proceedings of ACM SIGMOD*, Atlantic City, NJ (pp. 322-331).
- Berchtold, S., Keim, D., & Kriegel H.-P. (1996). The X-tree: An index structure for high-dimensional data. *Proceedings of VLDB*, Bombay, India.
- Bozkaya, T., Yazdani, N., & Ozsoyoglu, M. (1997). Matching and indexing sequences of different lengths. *Proceedings of CIKM*, Las Vegas, Nevada.
- Chan, K., & Fu, A.W. (1999). Efficient time series matching by wavelets. *Proceedings of IEEE ICDE* (pp. 201-208).
- Chandrasekaran, S., & Franklin, M.J. (2002). Streaming queries over streaming data. *Proceedings of VLDB*, Hong Kong, China.
- Faloutsos, C., Ranganathan, M., & Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. *Proceedings of ACM SIGMOD*, Minneapolis, Minnesota (pp. 419-429).

Gao, L., Yao, Z., & Wang, X.S. (2002c). Evaluating continuous nearest neighbor queries for streaming time series via pre-fetching. *Proceedings of VLDB*, Hong Kong, China.

Gao, L., & Wang, X.S. (2002a). Continually evaluating similarity-based pattern queries on a streaming time series. *Proceedings of ACM SIGMOD*, Madison, Wisconsin.

Gao, L., & Wang, X.S. (2002b). Improving the performance of continuous queries on fast data streams: Time series case. *Proceedings of SIGMOD/DMKD Workshop*, Madison, Wisconsin.

Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., & Strauss, M.J. (2003). One-pass wavelet decompositions of data streams. *IEEE Transactions on Knowledge and Data Engineering*, 15(3), 541-554.

Guha, S., Meyerson, A., Mishra, N., Motwani, R., & O'Callaghan, L. (2003). Clustering data streams: Theory and practice. *IEEE Transactions on Knowledge and Data Engineering*, 15(3), 515-528.

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. *Proceedings of ACM SIGMOD* (pp. 47-57). Boston.

Kontaki, M., & Papadopoulos, A.N. (2004). Similarity search in streaming time sequences. *Proceedings of SSDBM (to appear)*, Santorini, Greece.

Lin, K., Jagadish, H.V., & Faloutsos, C. (1995). The TV-tree: An index structure for high dimensional data. *The VLDB Journal*, 3, 517-542.

Liu, X., & Ferhatosmanoglu, H. (2003). Efficient k-NN search on streaming data series. *Proceedings of SSTD*, Santorini, Greece.

Park, S., Chu, W.W., Yoon, J., & Hsu, C. (2000). Efficient searches for similar subsequences of different lengths in sequence databases. *Proceedings of IEEE ICDE*.

Vlachos, M., Hatjieleftheriou, M., Gunopoulos, D., & Keogh, E. (2003). Indexing multidimensional time series with support for multiple distance measures. *Proceedings of ACM SIGKDD*, Washington, DC.

Weber, R., Schek, H.-J., & Blott, S. (1998). A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. *Proceedings of VLDB*, New York (pp. 194-205).

Yi, B.-K., Jagadish, H.V., & Faloutsos, C. (1998). Efficient retrieval of similar time sequences under time wrapping. *Proceedings of IEEE ICDE*, Orlando, Florida (pp. 201-208).

Yi, B.-K., & Faloutsos, C. (2000). Fast time sequence indexing for arbitrary Lp norms. *Proceedings of VLDB*, Cairo, Egypt.

## KEY TERMS

**Data Mining:** A research field which investigates the extraction of useful knowledge from large datasets. Clustering and association rule mining are two examples of data mining techniques.

**Dimensionality Reduction:** It is a technique that is used to lower the dimensionality of the original dataset. Each object is transformed to another object which is described by less information. It is very useful for indexing purposes, since it increases the speed of the filtering step.

**Distance Function:** It is used to express the similarity between two objects. It is usually normalized in the range between 0 to 1. Examples of distance functions used for time series data are the Euclidean distance and the Time Warping distance.

**Filter-Refinement Processing:** A technique used in query processing, which is composed of the filter step and the refinement step. The filter step discards parts of the database that cannot contribute to the final answer and determines a set of candidate objects, which are then processed by the refinement step. Filtering is usually enhanced by efficient indexing schemes for improved performance.

**Similarity Queries:** These are queries that retrieve objects which are similar to a query object. There are three basic similarity query types, namely, similarity range, similarity nearest-neighbor, and similarity join.

**Streaming Time Series:** It is composed of a sequence of values, where each value corresponds to a time instance. The length changes, since new values are appended.

**Time Series:** It is composed of a sequence of values, where each value corresponds to a time instance. The length remains constant.



# Spatio–Temporal Indexing Techniques<sup>1</sup>

**Michael Vassilakopoulos**  
*TEI of Thessaloniki, Greece*

**Antonio Corral**  
*University of Almeria, Spain*

## INTRODUCTION

Time and space are ubiquitous aspects of reality. Temporal and spatial information appear together in many everyday activities, and many information systems of modern life should be able to handle such information. For example, information systems for traffic control, fleet management, environmental management, military applications, local and public administration, and academic institutions need to manage information with spatial characteristics that change over time, or in other words, spatio-temporal information. The need for spatio-temporal applications has been strengthened by recent developments in mobile telephony technology, mobile computing, positioning technology, and the evolution of the World Wide Web.

Research and technology that aim at the development of Database Management Systems (DBMSs) that can handle spatial, temporal, and spatio-temporal information have been developed over the last few decades. The embedding of spatio-temporal capabilities in DBMSs and GISs is a hot research area that will continue to attract researchers and the informatics industry in the years to come.

In spatio-temporal applications, many sorts of spatio-temporal information appear. For example, an area covered by an evolving storm, the changing population of the suburbs of a city, the changing coast lines caused by ebb and tide. However, one sort of spatio-temporal information is quite common (and in some respects easier to study) and has attracted the most research efforts: moving objects or points, for example, a moving vehicle, an aircraft, a wandering animal.

One key issue for the development of an efficient spatio-temporal DBMS (STDBMS) is the use of spatio-temporal access methods at the physical level of the DBMS. The efficient storage, retrieval, and querying of spatio-temporal information demands the use of specialized indexing techniques that minimize the cost during management of such information.

In this article, we report on the research efforts that have addressed the indexing of moving points and other spatio-temporal information. Moreover, we discuss the

possible research trends within this area of rising importance.

## BACKGROUND

The term *spatial data* refers to multidimensional data, like points, line segments, regions, polygons, volumes, or other kinds of geometric entities, while the term *temporal data* refers to data varying in the course of time. Since in database applications the amount of data that should be maintained is too large for main memory, external memory (hard disk) is considered as a storage means. Specialized access methods are used to index disk pages and, in most cases, have the form of a tree. Numerous indexing techniques have been proposed for the maintenance of spatial and temporal data. Two good sources of related information are the survey by Gaede and Günther (1998) and the survey by Saltzberg and Tsotras (1999) for spatial and temporal access methods, respectively.

During last years, several researchers have focused on spatio-temporal data (spatial data that vary in the course of time) and the related indexing methods for answering spatio-temporal queries. A spatio-temporal query is a query that retrieves data according to a set of spatial and temporal relationships. For example, “find the vehicles that will be in a distance of less than 5km from a specified point within the next 5 minutes”. A number of recent short reviews that summarize such indexing techniques (especially, indexing of moving points) have already appeared in the literature. There are several ways for categorizing (several viewpoints, or classifications of) spatio-temporal access methods. In the rest of this section, we report on the approach followed by each of these reviews and on the material that the interested reader would find there.

In the book *Spatiotemporal Databases: The ChoroChronos Approach* that was authored within the ChoroChronos project and edited by Sellis et al. (2003), Chapter 6 is entitled “Access Methods and Query Processing Techniques” and reviews spatio-temporal access methods that have appeared up to 2001. The main classification followed in this chapter is between methods



belonging in the R-tree family and methods belonging in the Quadtree family. The principle guiding the hierarchical decomposition of data distinguishes between these two indexing approaches. The two fundamental principles, or hierarchies are:

- the data space hierarchy: a region containing data is split (when, for example, a maximum capacity is exceeded) to sub-regions in a way that depends on these data (for example, each of two sub-regions contains half of the data), and
- the embedding space hierarchy: a region containing data is split (when a certain criterion holds) to sub-regions in a predefined way (for example, a square region is always split in four quadrant sub-regions).

R-trees are data-driven, while Quadtrees are space-driven access methods.

The June 2002 (Vol. 25, No. 2) issue of the *IEEE Data Engineering Bulletin* (<http://sites.computer.org/debull/A02june/issue1.htm>) is devoted to “Indexing of Moving Objects” and is an excellent source of updated information. Pfoser (2002) reviews techniques that index the trajectories of moving objects that follow unconstrained movement (e.g., vessels at sea), constrained movement (e.g., pedestrians), and movement in transportation networks (e.g., trains or cars). These techniques allow the answering of queries concerning the past of the objects.

On the other hand, Papadopoulos et al. (2002) review techniques that index mobile objects and allow answering of queries about their future positions. The basis of these techniques is the duality transform, that is, mapping of data from one data space to another data space, where answering of queries is easier, or more efficient (mapping of a line segment representing a trajectory to a point in space of equal dimensionality).

Agarwal and Procopiu (2002) classify indexing techniques according to the consideration of time as another dimension (called time oblivious approach that can be used for answering queries about the past), the use of kinetic data structures (that can be used for answering present queries or even queries that arrive in chronological order), and the combined use of the two techniques (that can be used for efficiently answering queries about the near past or future).

Jensen and Saltenis (2002) discuss a number of techniques that may lead to improved update performance of moving-objects indices. Their paper is a good source of possible future research trends.

Chon, Agrawal, and El Abbadi (2002) report on managing object trajectories by following a partitioning approach that is best suited to answering time-dependent shortest path queries (where the cost of edges varies with time).

Mokbel, Ghanem, and Aref (2003) review numerous spatio-temporal access methods classifying them according to their ability to index only the past, only the present, and the present together with the future status of data. A very descriptive figure that displays the evolution of spatio-temporal access methods with the underlying spatial and temporal structures is included.

Tzouramanis, Vassilakopoulos, and Manolopoulos (2004) review and compare four temporal extensions of the Linear Region Quadtree and can store and manipulate consecutive raster images and answer spatio-temporal queries referring to the past.

## MAIN TRUST OF THE ARTICLE

In this section, we briefly review the most fundamental spatio-temporal indexing techniques.

### Quadtree-Based Methods

The Quadtree is a four-way tree where each node corresponds to a subquadrant of the quadrant of each father node (the root corresponds to the whole space). These trees subdivide space in a hierarchical and regular fashion. They are mainly designed for main memory; however, several alternatives for secondary memory have been proposed. The most widely used Quadtree is the Region Quadtree that stores regional data in the form of raster images. More details appear in Gaede and Günther (1998).

Tayeb, Ulusoy, and Wolfson (1998) used the PMR-quadtree for indexing future trajectories of moving objects. The PMR tree is a tree based on quadtrees, capable of indexing line segments. The internal part of the tree consists of an ordinary region quadtree residing in main memory. The leaf nodes of this quadtree point to the bucket pages that hold the actual line segments and reside on disk. Each line segment is stored in every bucket whose quadrant (region) it crosses. This causes the problem that data replication is introduced (every trajectory, a semi-infinite line, is stored in all the quadrants that it crosses).

Raptopoulou, Vassilakopoulos, and Manolopoulos (2004) used a new Quadtree-based structure, called XBR tree, for indexing past trajectories of moving objects. XBR trees (External Balanced Regular trees) are secondary memory balanced structures that subdivide space in a quadtree manner into disjoint regions. In their paper, XBR trees are shown to excel over PMR trees when used for indexing past trajectories.

Tzouramanis et al. (2003, 2004) present and compare four different extensions of the Linear Region Quadtree (the Time-Split Linear Quadtree, the Multiversion Linear Quadtree, the Multiversion Access Structure for Evolv-

ing Raster Images, and Overlapping Linear Quadtrees) for indexing a sequence of evolving raster data (e.g., images of clouds as they evolve in the course of time). A Linear Region Quadtree is an external memory version of the Region Quadtree, where each quadrant is represented by a codeword stored in a B+-tree. In their paper, temporal window queries (e.g., find the regions intersecting the query window within a time interval) are studied.

## R-Tree-Based Methods

An R-tree is a balanced multiway tree for secondary storage, where each node is related to a Minimum Bounding Rectangle (MBR), the minimum rectangle that bounds the data elements contained in the node. The MBR of the root bounds all the data stored in the tree. The most widely used R-tree is the R\*-tree. More details appear in Gaede and Günther (1998).

Xu, Han, and Lu (1990) presented an R-tree variation called RT-tree. The RT-tree couples time intervals with spatial ranges in each node of the tree: each MBR is accompanied by the time interval during which the related object or node is valid. Queries that embed time may require traversal of the whole tree.

In the same paper, the MR-tree was presented. This R-tree variation employs the idea of overlapping between a sequence of trees (like Overlapping Linear Quadtrees mentioned above) by storing common subtrees between consecutive trees only once. This tree suffers from reduced performance for temporal window queries. Additionally, a small change in a common subtree causes replication of this subtree between consecutive trees. Nascimento and Silva (1998) presented the HR-tree that is very similar to the MR-tree. Tao and Papadias (2001a) presented an improved version called the HR+-tree that avoids replication of subtrees to some extent. In the HR+-tree, a node is allowed to have multiple parents.

Theodoridis, Vazirgiannis, and Sellis (1996) presented the 3D R-tree that treats time as one of the three dimensions. Nascimento, Silva, and Theodoridis (1999) presented the 2+3 R-tree: a 2D R-tree is used for current 2 dimensional points and a 3D R-tree for the historical 3D trajectories. Depending on the query time, both trees may need to be searched.

Tao and Papadias (2001b) presented the MV3R-tree. This consists of a Multiversion R-tree (like the Multiversion Linear Quadtree mentioned above) to process timestamp queries and a 3D R-tree to process long interval queries.

Pfoser, Jensen, and Theodoridis (2000) presented the STR-tree (Spatio-Temporal R-tree), an R-tree with a different insert/split algorithm. This algorithm aims at keeping the line segments belonging to the same trajectory together as much as possible. In the same paper, the TB-tree,

Trajectory-bundle tree, was introduced. This is an R-tree-like structure that strictly preserves trajectories. A leaf node can only contain segments belonging to the same trajectory. The TB-tree outperforms the STR-tree in trajectory-based queries (queries involving the topology of trajectories and derived information, such as speed and heading of objects).

Hadjieleftheriou et al. (2002) use the partially-persistent R-tree (PPR-tree) to index general spatio-temporal data (not necessarily point data) that are allowed to move/change with a general motion over time. Two problems that arise are excessive dead space and overlap. These problems are overcome by making use of artificial updates.

Saltanis et al. (2000) proposed the TPR-tree (Time Parameterized R-tree) for indexing the current and future positions of moving points. This tree introduced the idea of parametric bounding rectangles in the R-tree (bounding rectangles that expand according to the maximum and minimum velocities of the objects that they enclose). To avoid the case where the bounding rectangles grow to be very large, whenever the position of an object is updated, all the bounding rectangles on the nodes along the path to the leaf at which this object is stored are recomputed. Tao, Papadias, and Sun (2003) presented an improvement called TPR\*-tree that uses new insertion and deletion algorithms that aim at minimizing a certain cost function. Moreover, Tao et al. (2004) introduced the STP tree, which is a generalization of the previous two trees to arbitrary polynomial functions describing the movement of objects.

Frentzos (2003) proposed the Fixed Network R-tree (FNR-tree) for indexing objects moving on fixed networks. The network is represented by a set of line segments (links) stored in a 2D R-tree. This tree is accompanied by a forest of 1D R-trees. There exists one 1D R-tree for each leaf node of the 2D R-tree. This 1D R-tree is used to index the time intervals that any moving object was moving on a link stored in this leaf.

## Transformation-Based Methods

Kollios, Gunopoulos, and Tsotras (1999) used the duality transformation to index the current and future positions of moving objects. A moving point in 1D space is represented by a trajectory (line segment) that is transformed to a point in the 2D space (accordingly, a moving point in 2D space is transformed to a point in 4D space). The resulting points are indexed by a kd-tree-based spatial index (a structure more suitable than R-trees, since the distribution of these points is highly skewed). A spatio-temporal range query is transformed into a polygon query in the dual space.

Agarwal, Arge, and Erickson (2000) also use a duality transformation. A moving object in the 2D space is represented by a trajectory in 3D space. This trajectory is projected into the (x, t) and (y, t) planes. The duals 2D points of each of these projections are indexed separately. The answer of a spatio-temporal range query is the union of two spatio-temporal range queries in the two planes. A kinetic data structure (Basch, Guibas & Hershberger, 1997) is used to index each dual space.

## FUTURE TRENDS

Apart from the development of new or improved indexing techniques, the following make up a non-exhaustive list of the research trends that are likely to be further addressed in the future.

- The evolution of indexing techniques for other sorts of spatio-temporal data (e.g., evolving boundaries, non-point objects that exhibit not only a changing position but rotate, change shape, color, transparency, etc.) apart from moving objects, or evolving regions.
- The development of access methods and query processing techniques for answering other kinds of queries (e.g., spatio-temporal joins) apart from range queries that have been mainly addressed so far.
- Efficiency and qualitative comparison of the indexing techniques proposed in the literature.
- The adoption of several of the update techniques proposed in Jensen and Saltenis (2002) like the utilization of buffering, the use of all available main memory, the consideration of movement constraints, and so forth.
- Consideration of distributed access methods for spatio-temporal data, as well as addressing concurrency and recovery issues and incorporation of uncertainty of movement, as proposed in Agarwal and Procopiuc (2002).

## CONCLUSION

In this article, we have reviewed the issues and techniques related to access methods for spatio-temporal data. This research area (and especially indexing of moving objects) has attracted many researchers during last years. Although this is a relatively new research area, numerous techniques have been developed. However, this is still a hot and demanding research area where many challenges need to be addressed.

## REFERENCES

- Agarwal, P.K., Arge, L., & Erickson, J. (2000). Indexing moving points. *Proceedings of the 19th ACM Symposium on Principles of Database Systems (PODS 2000)* (pp. 175-186).
- Agarwal, P.K., & Procopiuc, C.M. (2002). Advances in indexing for mobile objects. *IEEE Data Engineering Bulletin*, 25(2), 25-34.
- Basch, J., Guibas, L.J., & Hershberger, J. (1997). Data structures for mobile data. *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA 1997)* (pp. 747-756).
- Chon, H.D., Agrawal, D., & El Abbadi, A. (2002). Data management for moving objects. *IEEE Data Engineering Bulletin*, 25(2), 41-47.
- Frentzos, E. (2003). Indexing objects moving on fixed networks. *Proceedings of the 8th International Symposium on Spatial and Temporal Databases (SSTD 2003)* (pp. 289-305).
- Gaede, V., & Günther, O. (1998). Multidimensional access methods. *ACM Computing Surveys*, 30(2), 170-231.
- Hadjieleftheriou, M., Kollios, G., Tsotras, V.J., & Gunopoulos, D. (2002). Efficient indexing of spatiotemporal objects. *Proceedings of the 8th International Conference on Extending Database Technology (EDTB 2002)* (pp. 251-268).
- Jensen, C.S., & Saltenis, S. (2002). Towards increasingly update efficient moving-object indexing. *IEEE Data Engineering Bulletin*, 25(2), 35-40.
- Kollios, G., Gunopoulos, D., & Tsotras, V.J. (1999). On indexing mobile objects. *Proceedings of the 18th ACM Symposium on Principles of Database Systems (PODS 1999)* (pp. 261-272).
- Mokbel, M.F., Ghanem, T.M., & Aref, W.G. (2003). Spatio-temporal access method. *IEEE Data Engineering Bulletin*, 26(2), 40-49.
- Nascimento, M.A., Silva, J.R.O., & Theodoridis, Y. (1999). Evaluation of access structures for discretely moving points. *Proceedings of the International Workshop on Spatio-Temporal Database Management (STDBM 1999)* (pp. 171-188).
- Nascimento, M.A., & Silva, J.R.O. (1998). Towards historical R-trees. *Proceedings of the ACM Symposium on Applied Computing (SAC 1998)* (pp. 235-240).

Papadopoulos, D., Kollios, G., Gunopoulos, D., & Tsotras, V.J. (2002). Indexing mobile objects using duality transforms. *IEEE Data Engineering Bulletin*, 25(2), 18-24.

Pfoser, D. (2002). Indexing the trajectories of moving object. *IEEE Data Engineering Bulletin*, 25(2), 3-9.

Pfoser, D., Jensen, C., & Theodoridis, Y. (2000). Novel approaches to the indexing of moving object trajectories. *Proceedings of the 26th International Conference on Very Large Databases (VLDB 2000)* (pp. 189-200).

Raptopoulou, K., Vassilakopoulos, M., & Manolopoulos, Y. (2004). Towards Quadtree-based moving objects databases. *Proceedings of the 8th East-European Conference on Advances in Databases and Information Systems (ADBIS 2004)*, Budapest, Hungary (pp. 230-245).

Saltenis, S., Jensen, C.S., Leutenegger, S.T., & Lopez, M.A. (2000). Indexing the positions of continuously moving objects. *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2000)* (pp. 331-342).

Saltzberg, B., & Tsotras, V.J. (1999). Comparison of access methods for time-evolving data. *ACM Computing Surveys*, 31(2), 158-221.

Sellis, et al. (Eds.). (2003). Access methods and query processing techniques. *Spatiotemporal databases: The ChoroChronos approach* (pp. 169-217). Springer-Verlag.

Tao, Y., Faloutsos, C., Papadias, D., & Liu, B. (2004). Prediction and indexing of moving objects with unknown motion patterns. *Proceedings of the ACM Conference on the Management of Data (SIGMOD 2004)*, Paris (pp. 611-622).

Tao, Y., & Papadias, D. (2001a). Efficient historical R-trees. *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM 2001)* (pp. 223-232).

Tao, Y., & Papadias, D. (2001b). MV3R-Tree: A spatio-temporal access method for timestamp and interval queries. *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB 2001)* (pp. 431-440).

Tao, Y., Papadias, D., & Sun, J. (2003). The TPR\*-Tree: An optimized spatio-temporal access method for predictive queries. *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB 2003)* (pp. 790-801).

Tayeb, J., Ulusoy, O., & Wolfson, O. (1998). A Quadtree based dynamic attribute indexing method. *The Computer Journal*, 41(3), 185-200.

Theodoridis, Y., Vazirgiannis, M., & Sellis, T. (1996). Spatio-temporal indexing for large multimedia applications. *Proceedings of the 3rd IEEE International Conference on Multimedia Computing and Systems (ICMCS 1996)* (pp. 441-448).

Tzouramanis, T., Vassilakopoulos, M., & Manolopoulos, Y. (2003). Overlapping linear Quadtrees and spatio-temporal query processing. *The Computer Journal*, 43(4), 325-343.

Tzouramanis, T., Vassilakopoulos, M., & Manolopoulos, Y. (2004). Benchmarking access methods for time-evolving regional data. *Data & Knowledge Engineering*, 49(3), 243-286.

Xu, X., Han, J., & Lu, W. (1990). RT-Tree: An improved R-tree indexing structure for temporal spatial databases. *Proceedings of the International Symposium on Spatial Data Handling (SDH 1990)* (pp. 1040-1049).

## KEY TERMS

**Access Method or Indexing:** A technique of organizing data that allows the efficient retrieval of data according to a set of search criteria. R-trees and Quadtrees are two well-known families of such techniques.

**Constrained (Unconstrained) Movement:** Movement (of a moving object) that is (is not) confined according to a set of spatial restrictions.

**Movement in Transportation Networks:** Movement (of a moving object) that is confined on a transportation network (such as rails, or roads).

**Moving Object or Moving Point:** A data element that is characterized by its position in space that varies in the course of time (this is a kind of spatio-temporal datum).

**Spatio-Temporal Data:** Multidimensional data, like points, line segments, regions, polygons, volumes, or other kinds of geometric entities that vary in the course of time.

**Spatio-Temporal Database Management System:** A Database Management System that offers spatio-temporal data types and is able to store, index, and query spatio-temporal data.

**Spatio-Temporal Query:** A set of conditions embedding spatial and temporal relationships that define the set of spatio-temporal data to be retrieved.

**Trajectory:** The track followed by a moving object in the course of time (due to the change of its position).

## ENDNOTE

- <sup>1</sup> Supported by the ARCHIMEDES project 2.2.14, «Management of Moving Objects and the WWW», of the Technological Educational Institute of Thessaloniki (EPEAEK II), co-funded by the Greek Ministry of Education and Religious Affairs and the European Union.



# Storing XML Documents in Databases

**Albrecht Schmidt**

*Aalborg University, Denmark*

**Stefan Manegold**

*CWI, The Netherlands*

**Martin Kersten**

*Center for Mathematics and Computer Science, The Netherlands*

## INTRODUCTION

Ever since the Extensible Markup Language (XML) (W3C, 1998b) began to be used to exchange data between diverse sources, interest has grown in deploying data management technology to store and query XML documents. A number of approaches propose to adapt relational database technology to store and maintain XML documents (Deutsch, Fernandez & Suciu, 1999; Florescu & Kossmann, 1999; Klettke & Meyer, 2000; Shanmugasundaram et al., 1999; Tatarinov et al., 2002; O'Neil et al., 2004). The advantage is that the XML repository inherits all the power of mature relational technology like indexes and transaction management. For XML-enabled querying, a declarative query language (Chamberlin et al., 2001) is available.

Traditionally, database technology has been offering support for processing large amounts of data. Recent research has provided valuable insights into the nature of semistructured and XML data and has attempted to integrate them into existing paradigms. However, there are still challenges that have to be met to scale XML databases up to production levels as achieved by relational engines and, thus, to gain acceptance among practitioners. Naturally, XML warehouses inherit the power of relational warehouses (Roussopoulos, 1997), but they also face the same challenges; in particular, update and consistency problems of materialized, replicated, and aggregated views over source data need to be solved.

This article discusses techniques related to loading XML documents into a document warehouse. All techniques build on well-understood relational database technology and enable efficient management of large XML repositories. To get the most of relational database systems, we propose to do away with the pointer-chasing tree traversing operations, which many applications generate in the form of *edit scripts* and replace them with set-oriented operations. Edit scripts (Chawathe et al., 1996; Chawathe & Garcia-Molina, 1997) have been long known in text databases and are similar in behavior

to Document Object Model (DOM) (W3C, 1998a) traversals, which are standard in the XML world; they tend to put relational technology at a disadvantage due to their excessive use of pointer-chasing algorithms. We investigate the use of these scripts and propose alternative strategies for cases when they perform poorly.

We implemented our ideas in the XML extension of the Monet Database System (Schmidt, Kersten & Windhouwer, 2001; Schmidt et al., 2000). A more detailed description of our experiments is found in Schmidt and Kersten (2002). As we benchmarked the system's performance, it turns out that the use of edit-scripts is only sensible if they only update a rather small fraction of the database; once a certain threshold is exceeded, the replacement of a complete database segment is preferable. We discuss this threshold and try to quantify the trade-off for our example document database.

The application scenario which motivates our research consists of a set of XML data sources which are feature detectors that monitor multimedia data sources and analyze their content. The detectors feed protocols of analyses into a central data warehouse. The warehouse now provides the following services: (1) insertion of a documents (a data source transmits a single protocol of an analysis to the warehouse), (2) insertion of versioned sets of documents (a set of check-out points transmits the result of a bulk analysis transcript to the warehouse), (3) deletion of documents and sets of documents (a document is deleted from the warehouse because it has become invalid or stale; duplicate analyses and erroneous insertion also happen frequently and need to be corrected), and (4) execution of edit-scripts that are transmitted from the sources and systematically correct errors in already inserted documents; for example, *a posteriori* normalization of feature values is frequently required.

While we regard (1) as a special case of (2), hence, do not treat it separately, there is an obvious trade-off between a combination of (2) and (3) and the use of edit-scripts (4). More precisely, the question is: When is it

Figure 1. Example document

```
<image key="134" source="/cdrom/img1/293.jpeg">
  <date> 999010530 </date>
  <colors>
    <histogram> 0.399 0.277 0.344 </histogram>
    <saturation> 0.390 </saturation>
    <version> 0.8 </version>
  </colors>
</image>
```

cheaper to delete invalid data and reinsert a new consistent version than to use an edit script to “patch” the warehouse? This and other questions will be dealt with in detail later.

## BACKGROUND

XML documents are commonly represented as syntax trees. This section recalls some of the usual terminology we need to work with XML documents. In the sequel, *string* and *int* denote sets of character strings, respectively integers; *oid* denotes a set of unique object identifiers. Figure 1 shows an XML fragment, which is taken from the area of content-based multimedia retrieval (Schmidt & Kersten, 2002). Figure 2 displays the corresponding schema tree (dotted arrows indicate XML attribute relationships, straight lines XML element relationships).

Before we discuss techniques on how to store a tree as a database instance, we introduce the notion of *associations*. They are used to cluster semantically related information in a single relation and constitute the basis for the Monet XML Model; the aim of the clustering process is to enable efficient scans over semantically related data, that is, data with the same element ancestry, which are the physical backbone of declarative associative query language like SQL. Different types of

associations play different roles: associations of type *oid*×*oid* represent parent-child relationships. Both kinds of leaves, attribute values and character data, are modeled by associations of type *oid*×*string*, while associations of type *oid*×*int* are used to keep track of the original topology of a document. Paths describe the context of the element in the graph relative to the root node; we identify with *path(o)* the *type* of the association (×,*o*). The set of all paths in a document is called its *Path Summary*; it plays an important role in our query engine. The main rationale for the path-centric storage of documents is to evaluate the ubiquitous XML path expressions efficiently; the high degree of semantic clustering distinguishes our approach from other mappings (see Florescu & Kossmann, 1999 for a discussion). Our approach is to store all associations of the same “type” in one *binary relation*. A relation that contains the tuple (*·*,*o*) is named *R(path(o))*. In Figure 2, the types or paths are the *i*. Clustering XML elements by their type implies that we do not have to cope with many of the irregularities induced by the semi-structured nature of XML, which are typically taken care of with NULLs or overflow tables (Deutsch et al., 1999). In the sequel, we describe the machinery we need to convert documents to Monet format and bulkload them efficiently. Also note that we are able to reconstruct the original document given this path-centric representation. A detailed discussion of the reconstruction can be found in Schmidt et al. (2001). We remark that we can also access the documents in an object-oriented manner, that is, object as node in the syntax tree, which is often more intuitive to the user and is adopted by standards like the DOM (W3C, 1998a). However, we do not optimize for this as we see later.

## XML WAREHOUSES

### Populating the XML Warehouse

There are two basic notions of interest that we are going to discuss in this section as indicated in the introduction: populating a database from scratch, that is, bulk load, and incremental insertion of new data into an already existing database. However, similar technology underlies both cases. Let us consider an example first. There are two standard ways of accessing XML documents: (1) A low-level event-based, called SAX (Megginson, 2001), scans an XML document for token like start tag, end tag, character data, and so forth and invokes user-supplied functions for each token that is encountered in the input. The advantage of the SAX parsers is they only require minimal resources to work;

Figure 2. Schema tree of example document

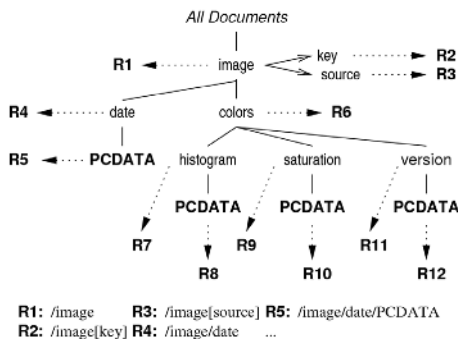


Figure 3. Path sequences in the example document

```

<image key="134" source="/cdrom/img1/293.jpeg">
<image><date>
<image><date>" 999010530 "
<image></date>
<image><colors>
<image><colors><histogram>
<image><colors><histogram>" 0.399 0.277 0.344 "
<image><colors></histogram>
<image><colors><saturation>
<image><colors><saturation>" 0.390 "
<image><colors></saturation>
<image><colors><version>
<image><colors><version>" 0.8 "
<image><colors></version>
<image></colors>
</image>

```

and (2) The more high-level DOM interface (W3C, 1998a) provides a standard interface to parse complete documents to syntax trees. In terms of resources, the memory consumption of DOM trees is much higher, linear in the size of the document; thus, it frequently happens that large documents exceed the size of available memory. We propose a bulk load method that has only slightly higher memory requirements than SAX-*O*(*height of document*)-but still keeps track of all the contextual information it needs. Thus, the memory requirements of the bulkload algorithm we use are very low as it does not materialize the complete syntax tree.

Since Monet XML stores complete paths, the bulk load routine has to track those paths. We do this by organizing the path summary as a schema tree which we use to efficiently map paths to relations. Each node in the schema tree represents a database relation and contains a tag name and reference to the relation. Figure 3 shows the path sequences generated by combining the SAX events of the parser and a stack in the following way. We attach OIDs to every tag when we put it on the stack. This way, we are able to record all path instances in the documents without having to maintain a syntax tree in (main) memory—an advantage that lets us process very large documents with relatively little memory. The function that performs the actual insertion is *insert(R,t)* where *R* is a reference to a relation and *t* is a tuple of the appropriate type. A first naive approach would thus result in the following sequence of insert statements (disregarding the order in the document and whitespace due to lack of space):

1. `insert(sys,(1,image))`
2. `insert(R(image[key]),(1,"134"))`
3. `insert(R(image[source]),(1,"/cdrom/img1/293.jpeg"))`
4. `insert(R(image/date),(1,2))`
5. `insert(R(image/date/pcdata),(2," 999010530 "))`
6. `insert(R(image/colors),(1,3))`

7. `insert(R(image/colors/histogram),(3,4))`
8. `insert(R(image/colors/histogram/pcdata),(4," 0.399 0.277 0.344 "))`
9. `insert(R(image/colors/saturation),(3,5))`
10. `insert(R(image/colors/saturation/pcdata),(5," 0.390 "))`
11. `insert(R(image/colors/version),(3,6))`
12. `insert(R(image/colors/version/pcdata),(6," 0.8 "))`

## Database Maintenance

Once data reside in a database, maintenance of these data becomes an important issue. We distinguish between two different maintenance tasks: First, the update of existing data via edit-scripts for propagating changes of source data to the warehouse, and, second, the deletion and insertion of complete versions of documents which may have become stale or need to be added to the warehouse.

The concept of edit-scripts to update hierarchically structured data is both intuitive and easy to implement on modern database systems (Chawathe et al., 1996; Chawathe & Garcia-Molina, 1997). The scripts comprise three basic operations for transforming the syntax tree: *insertion* of a node, *deletion* of node, and *moving* a node. (We do not mention other operators that traverse the syntax tree, see Chawathe & Garcia-Molina, 1997; Buneman et al., 1996). We also view these operations as representatives for traversals that are defined in the DOM standard (W3C, 1998a). Continuing our example, an edit script could insert additional subtrees that describe textures in the images or delete items that appear twice in the database. Typically, an edit-script first pins the location of nodes to be changed; this process is often done by navigating through the syntax tree as object identifiers in the database are often not accessible to other applications. Once the location is found, the scripts then apply update, delete, and insert operations. Conceptually, an edit-script may do two kinds of changes: *systematic* and *local* changes. Systematic changes may become necessary if a faulty application produced data with errors that are spread over parts of the XML document. In this case, the script traverses large parts of the syntax graph and applies similar changes at various places. In the relational context of our work, this may be an expensive restructuring process. On the other hand, if changes are only local, the script just visits a small number of nodes and patches them. This should be no resource-intensive problem, not in relational, object, or native systems.

We do not have the space to discuss edit-scripts in depth here and refer the reader to the above citations.

However, we demonstrate their use with an example similar to that used in the performance discussion. Consider again Figure 1. A systematic change would, for example, require us to change all dates from Unix system time, that is, seconds since January 1, 1970 to a more human readable format. The way we go about creating the appropriate edit-script is the following: We look up all associations which assign a value to an attribute *unit*. Then, for all these nodes, we calculate the new date and replace the old one. Techniques for constructing automata that do the traversal can be found, for example, in Neven and Schwentick (1999). Once such an automaton finds a node *n* that needs to be updated, it executes an *update(n, date, new date format)* statement. On the physical data model of the Background section, this is translated into a command that replaces the value of the respective association.

The point that is important for us is that edit-scripts traverse parts of the XML syntax graph and manipulate individual nodes. This is in stark contrast to the second method mentioned above, bulk deletion and reinsertion where we delete a complete segment of the database and reinsert a corrected version. In the example scenario, this means that an individual detector resends the corrected version of a previously submitted document instead of a patching edit-script. Generally, the underlying assumption is that the aforementioned data sources provide the capability of sending both, the edit-script and a complete updated document; however, this assumption holds for many practical applications as well as for our example: a detector may either send an edit-script or retransmit a corrected version of the complete document.

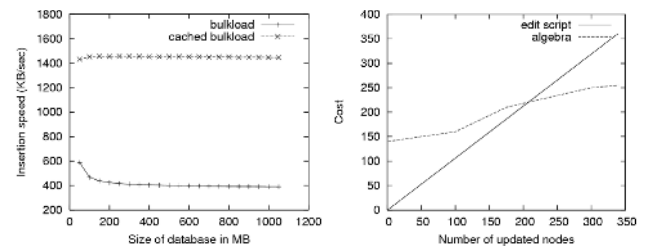
It is straightforward to design an algebraic algorithm that does the deletion and visits every node at most once like a linear scan (Schmidt & Kersten, 2002). Still, we need to discuss when to use bulk deletion combined with reinsertion and when to use edit-scripts. The next section looks quantitatively at when to go for what.

## Performance Impressions

This section presents performance impressions of a data warehouse (Kersten & Windhouwer, 2000) containing actual features which are more elaborate but similar to the ones used in the running example. The data warehouse uses Monet XML as the physical storage model. A 500 MHz Pentium-class PC running Redhat Linux was our experimentation platform.

Figure 4 shows the relationship between database size and insertion speed. The figure displays the speed-up of an optimized approach with caching over a naive implementation. As one might expect, the insertion into an empty database is faster than into an already densely

Figure 4. Performance of loading and updating data



populated one if no intelligent caching is used. As the database gets larger, insertion speed converges to a ratio of about 390 KB/sec. If schema trees are used, bulkload speed more than triples showing the potential of this technique, which has been explained in the Populating the XML Database section. Note that neither bulkload method blocks the database; both operate interactively and do not interfere with the transaction system. Note that the insertion performance in Figure 4 includes converting the textual representation of a document to executable database statements, thus, random memory accesses (which can be alleviated with path caching), whereas deletion can be done as sequential scans.

Updates are considered in Figure 4 as well. Edit-scripts, as presented in the Database Maintenance section, are run against the database created in the previous experiment; the updates they apply are systematic. The figure plots the performance of edit-scripts against a simple algebraic bulk replacement strategy which consists of deleting those documents that need updating from the database and subsequently loading updated documents into the database. With respect to when to choose which technique, the two lines in Figure 4 show that once more than approximately 220 entries are changed by the edit-script, one should consider reverting to bulk operations for performance reasons. The threshold of 220 entries is surprisingly low; however, one should keep in mind that relational databases are not optimized for pointer-chasing operations. We remark that this threshold also depends on the characteristics of the XML document, especially on the ratio between text and mark-up.

## Other Storage Mappings

This section discusses some opportunities and limitations of the methods discussed in this article. It takes up ideas found in the literature and relates them to the techniques we developed for Monet XML.



## Positional Mappings and Extent Mappings

Recently, the idea of using the OIDs located on a path as components of a positional number system like the Dewey system caught on (O'Neil et al., 2004; Tatarinov et al., 2002). We can easily adjust the mapping algorithms which produce the Monet mapping to include the whole list of OIDs which are on the stack rather than just include the two lowest ones as the algorithm in the Populating the XML Database section does. This just requires us to look at the complete stack before producing an insertion statement. This kind of positional mapping is especially suited for implementation of query primitives based on tree automata (Comon et al., 1997) since many of the query primitives like axis navigation can be translated transparently to state transitions (Ludäscher, Mukhopadhyay & Papakonstantinou, 2002).

Another type of mapping is called *extent mapping*. In extent mappings, not only the element OID is recorded but also the range in which the descendant node OIDs can be found; an example of such a mapping can be found in Zhang et al. (2001). The insertion algorithm can be adapted by delaying the generation of an insert statement until, not the start tag of an element, but its end tag is encountered. This way, enough information can be gathered about the position of the end tag.

## Semantic Mappings

A *semantic mapping* in our context is a mapping which takes into account data semantics. For example, a mapping might decide to map an XML element to an instance of a particular datatype. Semantic mappings generally follow two mutually exclusive principles. First, they may analyze schema information such as given by DTDs (Shanmugasundaram et al., 1999) or XML Schema and generate a database schema with features that imitate those found in E/R mappings. This kind of mapping is fully automatic. Second, users may manually specify application-specific constraints that help map document fragments to tuples in relational databases. These mappings try to overcome the semantic gap between XML and relational databases by equipping users with a language to specify which parts of a set of documents correspond to which part of a database schema.

Although semantic mappings are more complex to implement in a streaming manner like the Monet mapping where only a stack is needed to keep track of all necessary information, we can overcome much of the complexity by falling back to conceptually simpler mappings like the Monet mapping or a positional mapping. This is done by implementing the mapping as a two-

step process. In the first step, the Monet mapping of an incoming XML document is used to derive database relations. In the second step, these relations are combined in database queries to form the semantic entities produced by the semantic mapping. This abstraction can greatly facilitate the complexity of the mapping software.

## FUTURE TRENDS

In the future, we are likely to witness a tighter integration of the two paradigms which dominate XML processing: data management and information retrieval. This is likely to pose new challenges with respect to database maintenance and indexing; it is also likely to necessitate new data structures as well as novel query processing and update strategies and will thus require adaptations of the strategies presented in this article.

## CONCLUSION

This article discussed performance considerations for typical problems in relational XML document data warehousing, especially the trade-off between algebraic and pointer-chasing algorithms for updates. For practical purposes, it turned out that it often is better to replace a complete database segment and reinsert the updated data than to patch an existing version with expensive edit-scripts. In particular, our experiments showed that once the patched data volume exceeds a small percentage of the database, one should resort to bulk replacement. For good insertion performance, the use of schema trees has been beneficial.

## REFERENCES

- Buneman, P., Davidson, S.B., Hillebrand, G.G., & Suciu, D. (1996). A query language and optimization techniques for unstructured data. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Montreal, Canada (pp. 505-516).
- Chamberlin, D., Florescu, D., Robie, J., Siméon, J., & Stefanescu, M. (2001). XQuery: A query language for XML. Retrieved January 29, 2005, from <http://www.w3.org/TR/xquery>
- Chawathe, S., & Garcia-Molina, H. (1997). Meaningful change detection in structured data. *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 26-37).



## Storing XML Documents in Databases

- Chawathe, A., Rajaraman, A., Garcia-Molina, H., & Widom, J. (1996). Change detection in hierarchically structured information. *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 493-504).
- Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., & Tommasi, M. (1997). Tree automata techniques and applications. Retrieved January 29, 2005, from <http://www.grappa.univ-lille3.fr/tata>
- Deutsch, A., Fernandez, M.F., & Suci, D. (1999). Storing semistructured data with STORED. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Philadelphia (pp. 431-442).
- Florescu, D., & Kossmann, D. (1999). Storing and querying XML data using an RDBMS. *Data Engineering Bulletin*, 22(3).
- Kersten, M., & Windhouwer, M. (2000). DMW homepage. Retrieved January 29, 2005, from <http://www.cwi.nl/~acoi/DMW/>
- Klettke, M., & Meyer, H. (2000). XML and object-relational database systems - Enhancing structural mappings based on statistics. *International Workshop on the Web and Databases (WebDB)* (pp. 63-68).
- Ludäscher, B., Mukhopadhyay, P., & Papakonstantinou, Y. (2002). A transducer-based XML query processor. *Proceedings of the International Conference on Very Large Data Bases* (pp. 227-238).
- Megginson, D. (2001). SAX 2.0: The simple API for XML. Retrieved January 29, 2005, from <http://www.megginson.com/SAX/>
- Neven, F., & Schwentick, T. (1999). Query automata. *Proceedings of the 18th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (pp. 205-214).
- O'Neil, P., O'Neil, E., Pal, S., Cseri, I., Schaller, G., & Westbury, N. (2004). ORDPATHs: Insert-friendly XML node labels. Retrieved January 29, 2005, from <http://www.cs.umb.edu/~poneil/ordpath.pdf>
- Roussopoulos, N. (1997). Materialized views and data warehouses. *Proceedings of the 4th KRDB Workshop*. Retrieved January 29, 2005, from <http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-8/>
- Schmidt, A., & Kersten, M. (2002). Bulkloading and maintaining XML documents. *Proceedings of the ACM Symposium on Applied Computing (SAC)* (pp. 407-412).
- Schmidt, A., Kersten, M., & Windhouwer, M. (2001). Querying XML documents made easy: Nearest concept queries. *Proceedings of the IEEE International Conference on Data Engineering* (pp. 321-329).
- Schmidt, A., Kersten, M., Windhouwer, M., & Waas, F. (2000). Efficient relational storage and retrieval of XML documents. *International Workshop on the Web and Databases*, Dallas, Texas (pp. 47-52).
- Schmidt, A., Windhouwer, M., and Kersten, M. L. (1999). Feature Grammars. In *Proceeding of the International Conference on Information Systems Analysis and Synthesis*, Orlando, Florida.
- Shanmugasundaram, J., Tufte, K., He, G., Zhang, C., DeWitt, D., & Naughton, J. (1999). Relational databases for querying XML documents: Limitations and opportunities. *Proceedings of the International Conference on Very Large Data Bases*, Edinburgh, UK (pp. 302-314).
- Tatarinov, I., Viglas, S., Beyer, K., Shanmugasundaram, J., Shekita, E., & Zhang, C. (2002). Storing and querying ordered XML using a relational database system. *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 204-215).
- W3C. (1998a). Document object model (DOM). Retrieved January 29, 2005, from <http://www.w3.org/DOM/>
- W3C. (1998b). Extensible markup language (XML) 1.0. Retrieved January 29, 2005, from <http://www.w3.org/TR/1998/REC-xml-19980210>
- Zhang, C., Naughton, J., DeWitt, D., Luo, Q., & Lohman, G. (2001). On supporting containment queries in relational database management systems. *Proceedings of the ACM SIGMOD International Conference on Management of Data*.

## KEY TERMS

**Bulkload:** Adding a (large) set of data to a database rather than individual tuples.

**Database Maintenance:** The task of updating a database and enforcing constraints.

**Document Databases:** A collection of documents with a uniform user interface.

**Document Warehouses:** A document database consisting of documents gathered from various independent sources.

**Edit script:** A set of instructions that walk and update a document database node by node.

**Relational Databases:** Widely used variety of databases based on the relation algebra by Codd.

**XML:** The eXtensible Markup Language as defined at <http://www.w3.org/XML/>.

# Symbolic Objects and Symbolic Data Analysis

S

**Héctor Oscar Nigro**

*INCA/INTIA, Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina*

**Sandra Elizabeth González Císaro**

*Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina*

## INTRODUCTION

Today's technology allows storing vast quantities of information from different sources in nature. This information has missing values, nulls, internal variation, taxonomies, and rules. We need a new type of data that allow us to represent the complexity of reality, maintaining the internal variation and structure (Bock & Diday, 2000; Diday, 2002, 2003).

In Data Analysis Process or Data Mining, it is necessary to know the nature of null values<sup>1</sup>, also possible and valid to have some imprecision, due to differential semantics in a concept, diverse sources, linguistic imprecision, element resumed in database, human errors, and so forth (Chavent, 1997; Polaillon, 1998). Thus, we need a conceptual support to manipulate these types of situations. As we are going to see below, symbolic object constitutes a strong conceptual model.

A symbolic object is defined by its intent which contains a way to find its extent. For instance, the description of habitants in a region and the way of allocating an individual to this region is called intent, the set of individuals which satisfies this intent is called extent (Diday, 2003). For this type of analysis, different experts are needed, each one giving concepts.

Basically, Diday (2002) distinguishes between two types of concepts:

1. The concepts of the real world: Defined by an intent and an extent which exist, have existed, or will exist in the real world.
2. The *concepts of our mind* (among the so called *mental objects* by Changeux, 1983) which frame in our mind concepts of our imagination or of the real world by their properties and a "way of finding their extent" (by using the senses), and not the extent itself as undoubtedly, there is no room in our mind for all the possible extents (Diday, 2003).

A symbolic object models a concept in the same way our mind does by using a description "d" (representing its properties) and a mapping "a" able to compute its extent, for instance, the description of what we call a car

and a way of recognizing that a given entity in the real world is a car. Hence, whereas a concept is defined by intent and extent, it is modeled by intent and a way of finding its extent by symbolic objects like those in our mind. It should be noticed that it is quite impossible to obtain all the characteristic properties of a concept and its complete extent. Therefore, a symbolic object is just an approximation of a concept, and the problems of quality, robustness, and reliability of this approximation arise (Diday, 2003).

The theme is presented as follows: first, in the Background section, the history and fields of influence and sources of symbolic data. Second, in the main section, formal definitions of symbolic object, semantics applied to the symbolic object concept, and the methods are discussed. Third, future trends are given, followed by conclusions, references, and key terms.

## BACKGROUND

Diday presented the first article in 1988, in the Proceedings of the First Conference of the International Federation of Classification Societies (IFCS) (Bock & Diday, 2000). Then, much work has been done up to the publication of Bock and Diday (2000) and the Proceedings of IFCS'2000 (Bock & Diday, 2000; Diday 2003). Diday has directed an important quantity of his PhD thesis, with relevant theoretical aspects for symbolic object. Some of the most representative works are Brito (1991), De Carvalho (1992), Auriol (1995), Périnel (1996), Stéphan (1996), Ziani (1996), Chavent (1997), Polaillon (1998), Hillali (1998), Mfoummoune (1998), Vautrain (2000), Rodriguez Rojas (2000), De Reynies (2001), Vrac (2002), and Pak (2003).

Now, we are going to explain the fundamentals that symbolic objects hold from their fields of influence:<sup>2</sup>

- **Statistics:** From statistics, the symbolic object counts. It *knows* the distributions.
- **Exploratory Analysis:** The capacity of showing *new relations* between the descriptors {Tukey, Benzecri} (Bock & Diday, 2000).

- **Cognitive Sciences and Psychology:** The membership function of the symbolic object is to provide *prototypical instances* characterized by the most representative attributes and individuals {Rosch} (Diday, 2003).
- **Artificial Intelligence:** The representation of *complex knowledge* and the form of manipulation is more inspired from languages based on first order logic {Aristotle, Michalski, Auriol} (Diday, 2003).
- **Biology:** They use taxonomies and solutions widely investigated in this area {Adamson} (Bock & Diday, 2000).
- **Formal Concept Analysis:** The complete object symbolic is a Galois Lattice {Wille R.} (Bock & Diday, 2000; Polaillon, 1998).

In some of these sciences, the problem is how to obtain classes and their descriptions (Bock & Diday, 2000)

The *symbolic data tables* constitute the main input of a symbolic data analysis, helped by the *background knowledge* (Diday, 2003). In Chapter 1 of Bock and Diday’s book are mentioned as follow: each cell of this symbolic data table contains data of different types:

- Single quantitative value:** if « height » is a variable and w is an individual: height (w) = 3.5.
- Single categorical value:** Town (w) = Tandil.
- Multivalued:** in the quantitative case height (w) = {3.5, 2.1, 5} means that the height of w can be either 3.5 or 2.1 or 5.
- Interval:** height (w) = [3, 5], which means that the height of w varies in the interval [3, 5].
- Multivalued with weights:** for instance a histogram or a membership function (possibility, beliefs, or capacity theory can be also applied). Variables can be:
  - Taxonomic:** the color is considered to be light if it is white or pink.
  - Hierarchically dependent:** we can describe the kind of computers of a company only if it has a computer; hence, the variable “does the company have computers?” and the variable “kind of computers” are hierarchically linked.
  - With logical dependencies:** if age(w) is less than 2 months, then height (w) is less than 10.

**Example:** We need to model the concept of each airline company. We have stored Tables 1 and 2.

Symbolic data are generated when we summarize huge sets of data. The need for such a summary can appear in

Table 1. Travel

TYPE	COMPANY	WORKERS
T <sub>11</sub>	AEROLINEAS	4
T <sub>12</sub>	IBERIA	5
T <sub>13</sub>	AEROLINEAS	6
T <sub>14</sub>	IBERIA	3
T <sub>15</sub>	AIR FRANCE	7
T <sub>16</sub>	AIR FRANCE	3

Table 2. Flights

TRAVEL	COMPANY	TIMETA BLE	DESTINATION
V <sub>11</sub>	AEROLINEAS	12:30	NEW YORK
V <sub>12</sub>	IBERIA	15:40	BS. AS.
V <sub>13</sub>	AEROLINEAS	0:30	MADRID
V <sub>14</sub>	AIR FRANCE	10:30	PARIS
V <sub>15</sub>	AIR FRANCE	13:40	MIAMI
V <sub>16</sub>	AEROLINEAS	18:00	NEW YORK
V <sub>17</sub>	IBERIA	7:00	ROMA

Table 3. Symbolic table, modeling the concept of company

COMPANY	DESTINATION	TYPE	TRAVEL	WORKERS
AEROLINEAS	<sup>2</sup> / <sub>3</sub> NEW YORK,	T <sub>11</sub> , T <sub>13</sub>	V <sub>11</sub> , V <sub>13</sub> , V <sub>16</sub>	[4, 6]
	<sup>2</sup> / <sub>3</sub> MADRID			
IBERIA	<sup>1</sup> / <sub>2</sub> BS. AS,	T <sub>12</sub> , T <sub>14</sub>	V <sub>12</sub> , V <sub>17</sub>	[3, 5]
	<sup>1</sup> / <sub>2</sub> ROMA			
AIR FRANCE	<sup>1</sup> / <sub>2</sub> PARIS,	T <sub>15</sub> , T <sub>16</sub>	V <sub>14</sub> , V <sub>15</sub>	[3, 7]
	<sup>1</sup> / <sub>2</sub> MIAMI			

different ways, for instance, from any query to a database, which induces categories and descriptive variables. Symbolic data can also appear after a clustering in order to describe in an explanatory way (by using the initial variables) the obtained clusters (Diday, 2003; Bock & Diday, 2000).

Symbolic data may also be “native” in the sense that they result from *expertise knowledge* (type of emigration, species of insects), probable distribution, percentages, or range of any random variable associated with each cell of a stochastic data table, time series, confidential data, and so forth. Also, they result from relational databases in order to study a set of units whose description needs the merging of several relations (Diday & Billard, 2002).

## SYMBOLIC OBJECT AND SYMBOLIC DATA ANALYSIS

### Symbolic Object: Formal Definitions

A Symbolic Object is a triple  $s = (a, R, d)$  where  $R$  is a relation between descriptions,  $d$  is a description and “ $a$ ” is a mapping defined from  $\Omega$  in  $L$  depending on  $R$  and  $d$ . (Diday, 2003)

Most of the symbolic data analysis algorithms give in their output the symbolic description “ $d$ ” of a class of individuals by using a generalization process. By starting with this description, symbolic objects give a way to find, at least, the individuals of this class. Example: The age of two individuals  $w_1, w_2$  are  $\text{age}(w_1) = 30, \text{age}(w_2) = 35$ ; the description of the class  $C = \{w_1, w_2\}$  obtained by a generalization process can be  $[30, 35]$ . In this simple case, the symbolic object “ $s$ ” is defined by a triple:  $s = (a, R, d)$  where  $d = [30, 35]$ ,  $R = “\in”$  and “ $a$ ” is the mapping:  $\Omega \rightarrow \{\text{true}, \text{false}\}$  such that  $a(w) = \text{the true value of age}(w) R d$  denoted  $[\text{age}(w) R d]$ . An individual  $w$  is in the extent of  $s$  iff  $a(w) = \text{true}$  (Diday, 2002).

There are two kinds of symbolic objects:

- **Boolean Symbolic Objects:** The instance of one binary relation between the descriptor of the object and the definition domain that is defined to have values true or false. If  $[y(w) R d] = \{\text{true}, \text{false}\}$ , then it is a Boolean symbolic object (Diday, 2002, 2003). Example:  $s = (\text{color} \in \{\text{red}, \text{white}\})$ , here we are describing an individual /class whose color is red or white.
- **Modal Symbolic Objects:** In some situations, we can’t say true or false, we have a degree of belonging, or some linguistic imprecision as always true, often true, fifty-fifty, often false, always false; now we say that the relation is fuzzy. If  $[y(w) R d] \in L = [0,1]$  is a modal symbolic object (Diday, 2003; Diday & Billard, 2002). Example:  $s = (\text{color} \in [(0,25) \text{red}, (0,75) \text{white}])$ , at this point we are describing an individual/class that has color: 0,25 red; 0,75 white. This weighs can have different meaning, which are going explain to below.

Diday called *assertion* a special case of a symbolic object defined by  $s = (a, R, d)$  where  $R$  is defined by  $[d' R d] = \bigwedge_i i=1, p [d'_i R_i d_i]$  where “ $\wedge$ ” has the standard logical meaning and “ $a$ ” is defined by:  $a(w) = [y(w) R d]$  in the Boolean case. Notice that considering the expression  $a(w) = \bigwedge_i i=1, p [y_i(w) R_i d_i]$ , we are able to define the symbolic object  $s = (a, R, d)$  (Diday, 2002; Diday & Billard, 2002).

The *extension* of a symbolic object is a function that permits us recognize when an individual belongs the class description.

In the Boolean case, the extent of a symbolic object is denoted  $\text{Ext}(s)$  and defined by the extent of  $a$ , which is:  $\text{Extent}(a) = \{w \in \Omega / a(w) = \text{true}\}$ . In the modal instance, given a threshold  $\alpha$ , it is defined by  $\text{Ext}\alpha(s) = \text{Extent}\alpha(a) = \{w \in \Omega / a(w) \geq \alpha\}$  (Diday, 2002; Diday & Billard, 2002).

### Semantics Applied to the Symbolic Object Concept

The semantics are applied in the modal symbolic objects and are the weights observed in the previous example of this type of symbolic object (Bock & Diday, 2000):

- **Probabilities:** This is the classical situation; Kolmogorov’s axiom:

$$\Pr(E1 \cup E2) = \Pr(E1) + \Pr(E2) - \Pr(E1) \cap E2$$

- **Capacities:** At this point, there are groups with probabilistic multinomial variables. The capacity is interpreted in Choquet’s sense: “The probability of at least one individual belongs this value”, the capacity of a modality is the union capacity. Consequently, the capacity of  $SO_1$  and  $SO_2$  for  $M_1$  is  $p_{11} + p_{21} - p_{11} * p_{21}$ , and the capacity of  $SO_1, SO_2, \dots, SO_n$  for  $M_1$  is computed using the associative property.
- **Possibilities:** When the frequency theory fails. Example: We have a math book, but the word *mathematics* appears few times. The mathematician knows what this book is about. In this situation, we need different experts to give us their experience. So Zadeh’s axiom is used for fuzzy sets:

$$\text{Pos}(E1 \cup E2) = \text{Máx.}(\text{Pos}(E1), \text{Pos}(E2))$$

- **Beliefs:** Here, the expert says: “I think that the probability of the occurrence of  $A$  is  $p_1, p_2$  is  $B$ , but I do not know everything of  $A$  and  $B$ “. If  $p_1 + p_2 < 1$ , then  $1 - (p_1 + p_2)$  expresses his ignorance. Schaffer’s axiom (Bender, 1996):

$$\text{Bel}(E1 \cup E2) \geq \text{Bel}(E1) + \text{Bel}(E2) - \text{Bel}(E1) \cap E2$$

Dumpster’s rule allows combining the beliefs of various experts.

We can see that in these theories emphasis falls mainly on the knowledge of the expert. The first two



theories work with empirical information; the last ones have a higher level of information based on concepts.

### The Methods

In the article “An Introduction to Symbolic Data Analysis and the Sodas Software” Diday (2002) presents the main characteristics for the methods by the following principles:

1. They start as input with a symbolic data table, and they give as output a set of symbolic objects. These symbolic objects give explanation of the results in a language close to the one of the user.
2. They use efficient generalization processes during the algorithms in order to select the best variables and individuals.
3. They provide graphical descriptions taking into account the internal variation of the symbolic objects.

We prepared a table to give an idea about the methods implemented in Sodas<sup>3</sup>.

### FUTURE TRENDS

The most important features incorporated in software Sodas2<sup>4</sup> are: neuronal networks, regression, multidimensional scaling, mixture decomposition, Kohonen maps, and propagation of the obtained symbolic objects. Some of the most recent advances are mixture decomposition of distributions of distributions (by Copulas, Dirichlet, and Kraft, stochastic process), stochastic symbolic, conceptual lattices using capacity theory, symbolic class description and symbolic regression (Diday, 2002, 2004).

Future investigations and applications are oriented to spatial symbolic clustering by pyramids, symbolic time series, consensus between different descriptions of the same set of units (Diday, 2004).

### CONCLUSION

The need to extend standard data analysis methods (exploratory, clustering, factorial analysis, discrimination, and so forth) to symbolic data tables in order to extract new knowledge is increasing due to the expansion of information technology, now able to store an increasingly larger amount of huge data sets. This requirement has led to a new methodology called *symbolic data analysis* whose aim is to extend standard data analysis

methods to a new kind of data table, called *symbolic data table*, and to give more explanatory results expressed by real-world concepts mathematically represented by easily readable symbolic objects (Diday & Billard, 2002; Bock & Diday, 2000).

Some advantages are: It preserves the confidentiality of the information; supports the initial language in which they were created; allows the spread of concepts between databases. Independent from the initial table, they are capable of identifying some individual coincidence described in another table (Bock & Diday, 2000; Diday, 2003).

As a result of working with higher units, called concepts, necessarily described by more complex data extending data mining to knowledge mining (Diday, 2004).

The concept of symbolic object is very important for the construction of the data warehouse, and it is an important development for data mining, especially for the manipulation and analysis of aggregated information.

Table 4. Methods in Sodas software

Method's name in SODAS	Meaning	Authors of the theoretical aspects	References
<i>DBSO</i>	Creation of Symbolic Objects from Relational Databases	Stephan, Csernel	Bock & Diday, 2000; Hebrail & Lechevallier, 2000; Sodas Home page
<i>DI</i>	Measures of similarity, dissimilarity, matching for Boolean Symbolic Objects	Diday, Ichino, Gowda, Yaguchi, De Carvalho	Bock & Diday, 2000; Diday & Billard, 2002; Espósito, Malerba, Gioviale & Tamma, 2001; Sodas Home page
<i>DIM</i>	Measures of similarity and dissimilarity for Modal Symbolic Objects	Diday, Ichino, Gowda, Yaguchi, De Carvalho	Bock & Diday, 2000; Diday & Billard, 2002; Sodas Home page
<i>STAT</i>	Descriptive Statistics	Bertrand, Billard, Diday, Goupil	Bock & Diday, 2000; Diday & Billard, 2002; Sodas Home page
<i>PCA</i>	Principal Component Analysis	Chouakria, Diday, Cazes	Bock & Diday, 2000; Diday & Billard, 2002; Lauro & Palumbo, 2000; Sodas Home page
<i>FDA</i>	Factorial Discriminant Analysis	Diday, Lauro, Palumbo, Verde	Bock & Diday, 2000; Diday & Billard, 2002; Sodas Home page
<i>DSD</i>	Discriminant Symbolic Description	Gettler-Summa	Bock & Diday, 2000; Gettler-Summa, 2000; Sodas Home page
<i>PYR</i>	Symbolic Pyramid	Bertrand, Brito, Diday, Polaillon	Bock & Diday, 2000; Brito, 2001; Polaillon, 1998, Sodas Home page
<i>DIV</i>	Divisive Clustering	Chavent	Bock & Diday, 2000; Chavent, 1997, 1998; Sodas Home page
<i>SDT</i>	Strata Decision Tree	Bravo	Bock & Diday, 2000; 4, Sodas Home page.
<i>TREE</i>	Decision Tree	Périnel	Bock & Diday, 2000; Sodas Home page.
<i>DKS</i>	Discriminant Kernel Symbolic Analysis	Lissoir & Rasson	Bock & Diday, 2000; Sodas Home page.
<i>SOE</i>	Symbolic Object Editor	Noirhomme	Bock & Diday, 2000; Noirhomme, 2002, 2004; Sodas Home page.

## REFERENCES

- ASSO. Project home page. Retrieved July, 2004, from <http://www.info.fundp.ac.be/asso/>
- Auriol, E. (1995). *Intégration d'approches symboliques pour le raisonnement à partir d'exemples*. Thèse de Doctorat, Université Paris 9 Dauphine, France.
- Bender, E. (1996). *Mathematical methods in artificial intelligence*. IEEE Computer Society Press.
- Bock, H.H., & Diday, E. (2000). *Analysis of symbolic data. Studies in classification, data analysis and knowledge organization*. Springer Verlag.
- Bravo, C. (2002, July). *Strata decision tree: A symbolic data analysis technique*. Workshop Symbolic Data Analysis. Cracow University of Economics. CRACOW (Poland). Retrieved July, 2002, from <http://www.jsda.unina2.it/WorkSDA/BravoWSDAfinal.pdf>
- Brito, P. (1991). *Analyse de données symboliques. Pyramides de d'héritage*. Thèse de Doctorat, Université Paris 9 Dauphine, France.
- Brito, P. (2001). *Hierarchy and pyramid clustering*. ASSO Meeting Paris 2001. Retrieved August, 2002, from [http://www-rocq.inria.fr/sodas/WP6/WP6.1-Dauphine2001\\_fichiers/v3\\_document.htm](http://www-rocq.inria.fr/sodas/WP6/WP6.1-Dauphine2001_fichiers/v3_document.htm)
- Chavent, M. (1997). *Analyse des Données symboliques. Une méthode divisive de classification*. Thèse de doctorat in Sciences. l'Université Paris IX-Dauphine. Retrieved September, 2002, from <http://www.math.u-bordeaux.fr/~chavent/thesemc.ps>
- Chavent, M. (1998). *A monothetic clustering method*. Pattern recognition letters. 989-996. Retrieved August, 2002, from <http://www.math.u-bordeaux.fr/~chavent/PRL98.ps>
- De Carvalho, F. (1992). *Méthodes descriptives en analyse des données symboliques*. Thèse de Doctorat in Informatique des Organisations, Université Paris 9 Dauphine, France.
- De Reynies. (2002). *Classification et discrimination en analyse de données symboliques: Application à l'exploitation de scénarios d'accidents routiers*. Thèse de Doctorat, Université Paris 9 Dauphine, France.
- Diday, E. (2002). An introduction to symbolic data analysis and the Sodas software. *The Electronic Journal of Symbolic Data Analysis*, 0, 000-000. Retrieved Devember, 2002, 2005, from <http://www.jsda.unina2.it/volumes/Vol0/Edwin.pdf>
- Diday, E. (2002, July). *The future of symbolic data analysis*. Workshop Symbolic Data Analysis. Cracow University of Economics. CRACOW (Poland). Retrieved November 2002, from <http://www.jsda.unina2.it/WorkSDA/DidayWSDA.pdf>
- Diday, E. (2003, September 3-6). *Concepts and Galois lattices in symbolic data analysis*. Journées de l'Informatique Messine. Knowledge Discovery and Discrete Mathematics Metz, France. Retrieved October, 2003, from <http://www.inist.fr/uri/jim03/diday.pdf>
- Diday, E. (2004, January). From data mining to knowledge mining: Symbolic data analysis and the Sodas software. *Proceedings of the Workshop on Applications of Symbolic Data Analysis*, Lisboa, Portugal. Retrieved November, 2003, from <http://www.info.fundp.ac.be/asso/dissem/W-ASSO-Lisbon-Intro.pdf>
- Diday, E., & Billard, L. (2002). Symbolic data analysis: Definitions and examples. Retrieved May, 2004, from [http://www.stat.uga.edu/faculty/LYNNE/tr\\_symbolic.pdf](http://www.stat.uga.edu/faculty/LYNNE/tr_symbolic.pdf)
- Espósito, F., Malerba, D., Gioviale, V., & Tamma, V. (2001). Comparing dissimilarity measures for symbolic data analysis. *Proceedings of the Joint Conferences on New Techniques and Technologies for Statistics and Exchange of Technologies and Know-How* (pp. 473-481). Retrieved November, 2002, from <http://www.di.uniba.it/~malerba/publications/ntts-asso.pdf>
- Gettler-Summa, M. (2000). Marking and generalization by symbolic objects in the symbolic official data analysis software. *Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases*, Lyon, France, September. Retrieved January, 2003, from [http://eric.univ-lyon2.fr/~pkdd2000/Download/WS6\\_8.pdf](http://eric.univ-lyon2.fr/~pkdd2000/Download/WS6_8.pdf)
- Hebrail, G., & Lechevallier, Y. (2000). DB2SO: A software for building symbolic objects from databases. Retrieved October, 2002, from <http://www-rocq.inria.fr/axis/papers/00smpa/hebrail.pdf>
- Hillali Y. (1998). *Analyse et modélisation des données probabilistes: Capacités et lois multidimensionnelles*. Thèse de Doctorat, Université Paris 9 Dauphine, France
- Lauro, C., & Palumbo (2000). Principal component analysis of interval data: A symbolic data analysis approach. *Computational Statistics*, 15(1), 73-87.
- Mfoummoune, E. (1998). *Aspects algorithmiques de la classification ascendante pyramidale*. Thèse de Doctorat, Université Paris 9 Dauphine, France.

Noirhomme, M. (2002). Visualization of large data sets: The zoom star solution. *The Electronic Journal of Symbolic Data Analysis*, 0, 000-000. Retrieved December, 2002, from <http://www.jsda.unina2.it/volumes/Vol0/noirho.pdf>

Noirhomme, M. (2004). Visualisation of symbolic data. *Proceedings of the Workshop on Applications of Symbolic Data Analysis*, Lisboa, Portugal. Retrieved May, 2004, from <http://www.info.fundp.ac.be/asso/dissem/W-ASSO-Lisbon-Visu.pdf>

Pak, K. (2003). *Interprétation des pyramides*. Thèse de DEA. Université Paris 9 Dauphine, France.

Périnel, E. (1996). *Segmentation et Analyse des Données Symboliques: Application à des données probabilistes imprécises*. Thèse de Doctorat, Université Paris 9 Dauphine, France.

Polailon, G. (1998). Organisation et interprétation par les treilles de Gallois de données de type multivalué, intervalle ou histogramme. Thèse Docteur in Informatique. l'Université Paris IX-Dauphine. Retrieved August, 2005, from <http://wwwsi.supelec.fr/gp/these.ps.gz>

Touati, M., & Diday, E. SODAS home page. Retrieved July, 2004, from <http://www.ceremade.dauphine.fr/~touati/sodas-pagegarde.htm>

## KEY TERMS

**Artificial Intelligence:** The field of science that studies how to make computers “intelligent”. It consists mainly of the fields of machine learning (neural networks and decision trees) and expert systems. The principal problem is how to represent knowledge.

**Decision Trees:** A method of finding rules or *rule induction* which divide the data into subgroups that are as similar as possible with regard to a target variable (variable that we want to explain).

**Exploratory Analysis:** It is part of the Data Analysis French School, developed among 1960 and 1980. The principal authors are Tuckey and Benzecri. The process of analysis takes as a target to discover new relations between the sets of the analyzed information.

**Extension:** “I call the extension of an idea the subjects to which it applies, which are also called the inferiors of a universal term, that being called superior to them” (Arnault & Nicole, 1662).

**Formal Analysis Concept:** A theory of data analysis, which identifies conceptual structures among data sets; Rudolf Wille introduced it in 1982. It structures data into units that are formal abstractions of concepts of human thought, allowing meaningful and comprehensible interpretation. FCA models the world as composed of objects and attributes. A strong feature of formal concept analysis is its capability of producing graphical visualizations of the inherent structures among data. In the field of information science, there is a further application: the mathematical lattices that are used in formal concept analysis can be interpreted as classification systems. Formalized classification systems can be analyzed according to the consistency of their relations. (FAC Home Page <http://www.upriss.org.uk/fca/fca.html>)

**Fuzzy Sets:** Let  $U$  be a set of objects so called universe of discourse. A fuzzy set  $F$  in  $U$  is characterized by a function of inclusion  $m_F$  taking values in the interval  $[0,1]$ , i.e.  $\mu_F : U \rightarrow [0,1]$ ; where  $\mu_F(u)$  represents the degree in which  $u \in U$  belongs to fuzzy set  $F$ .

**Galois Lattice:** Galois Lattice provides some meanings to analyze and represent data. This refers to two-ordered set. An ordered set  $(I, \#)$  is the set  $I$  together with a partial ordering  $\#$  on  $I$ .

**Intension:** This is the comprehension of an idea. “I call the comprehension of an idea the attributes which it contains and which cannot be taken away from it without destroying it” (Arnault & Nicole, 1662).

## ENDNOTES

- <sup>1</sup> The cases are by absence value, null value, or default value.
- <sup>2</sup> Here, we name between { } the principals authors; Diday referenced those.
- <sup>3</sup> This software is the result of the investigation in which 17 institutions from 9 European countries are concerned and three official statistical institutions were involved in this project EUSTAT (Spain), INE (Portugal), and ONS (England). ASSO project continues investigating and developing new functionality for the software.
- <sup>4</sup> Recently, this software was presented in the International Workshop on Symbolic Data Analysis on 6-7 May 2004 at PARIS-IX University Dauphine, France.

# Syntactical and Semantical Correctness of Pictorial Queries for GIS

**Fernando Ferri**

*Instituto di Ricerche sulla Popolazione e le Politiche Sociali, Italy*

**Maurizio Rafanelli**

*Instituto di Analisi dei Sistemi ed Informatica "A. Ruberti", Italy*

## INTRODUCTION

One of the main topics in geographical information systems (GIS) research concerns the definition of high level visual query languages (Chrisman, 2002; Laurini & Thompson, 1992). This arises from the need to provide the user with a visual interactive tool for data manipulation and retrieval that is independent of the data's physical organization. The use of standard query languages for spatial data handling (Rigaux, Scholl, & Voisard, 2001; Shekhar et al. 1999) has been hindered by the lack of appropriate language support. In fact, in visual query languages for GIS, a query can lead to multiple interpretations (Favetta & Aufaure-Portier, 2000).

For example, suppose the user wishes to formulate the following query: "Find all the regions that are *passed through* by a river and *overlap* a forest." In this query, the user does not express interest in the relationship between the river and the forest. However, when he or she draws a shape representing a region, and another shape representing a river, he or she cannot avoid representing a spatial relationship between them, and so every representation considering a specific relationship between the two features can be considered a valid representation of the query. Different visual queries can thus represent the previous query in natural language. In Figure 1.a, the river passes through the forest; in Figure 1.b, the river touches the forest; and in Figure 1.c, the forest and the river are "disjointed."

Thus, the three representations should be interpreted as three different queries and, moreover, each query has a different meaning from the original query in natural

language. Owing to semantical ambiguity problems, some configuration could be semantically invalid. For example, a lake cannot include a region. However, a visual query language with clear syntax and semantics can a priori overcome many cases of the ambiguities, minimizing multiple interpretations of a query for both the system and the user. This article discusses the syntactic and semantic correctness of spatial configurations in the context of nonprocedural geographic pictorial query languages. Thus, this article considers possible ambiguities related to visual representations of a query, and it does not consider ambiguities related to interactions between system and user.

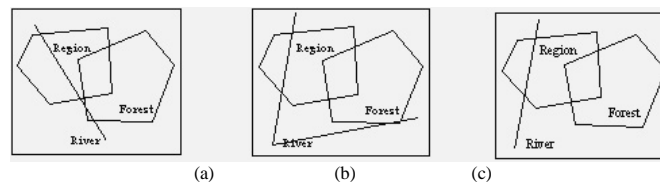
## BACKGROUND

Various proposals of visual query languages for geographical data have recently been made. It is possible to classify these different languages into two main approaches.

In the first approach, the user draws his or her query freehand, directly on the screen, using the blackboard metaphor. Examples of this are *Sketch* (Meyer, 1993) and *Spatial-Query-by-Sketch* (Blaser & Egenhofer, 2000; Egenhofer, 1997). With this approach, the parser considers both the exact solution of the query and other approximate solutions obtained by removing or relaxing some constraints.

In the second approach, the user is free to draw iconic symbols on the screen, to express an object or an operator. Important examples of this approach are the

Figure 1. Three visual queries representing the same query expressed in natural language





Cigales language (Calcinelli & Mainguenaud, 1994; Mainguenaud & Portier, 1990) and the Lvis language (Aufaures-Portier, 1995; Aufaures-Portier & Bonhomme, 1999). In particular, Cigales is based on the idea of expressing a query by drawing the pattern corresponding to the result the user desires. The graphical forms and icons that conceptualize the operators are predefined.

Lvis is an extension of Cigales. The most relevant difference consists of the definition of new operators, because both spatial and temporal properties of the objects forming the query are considered. For this language, too, the main limitation is that there are different interpretations of the same query, that is, with the same visual representation the system is not able to give a unique interpretation. Ambiguity is increased with an increasing number of query objects.

Favetta and Aufaure-Portier (2000) confronted this problem and proposed a taxonomy based on user actions and system materialization (i.e., the different images the system can materialize), distinguishing ambiguities in visual query languages for GIS and ways to resolve them. They also proposed a model to solve a particular case of ambiguity. The proposed system, an enlargement of Lvis, established a dialogue with the user. Whenever an ambiguity occurred, it showed all the available configurations and requested a choice. The authors concluded that the strategy for avoiding ambiguities in most visual geographic query languages was to define not fully visual, but hybrid languages, including a textual part, and to offer a grammar with low expressive power.

The geographical pictorial query language (GeoPQL; Ferri & Rafanelli, 2004) is an evolution of the pictorial query language (PQL; Ferri, Massari, & Rafanelli, 1999), which resolved some previous limitations on ambiguities. It is possible to specify queries using *symbolic graphical objects* (SGOs) that have the appearance of the three classic shapes: point, polyline, and polygon. It is possible to assign to each SGO a semantic linked to the different kinds of information (i.e., layer) in the geographical database and impose constraints on both the SGO's attributes and its spatial position. In addition, GeoPQL uses a limited set of graphical symbols to represent some operators that do not have a representation expressing the involved relations. Consequently, a generic pictorial sentence is represented by a set of symbolic graphical objects, a set of possible properties of each SGO, a set of possible symbolic operators, and the target of the query.

The GeoPQL algebra consists of two geometric operators: geo-union (Uni) and geo-difference (Dif); nine topological operators: geo-disjunction (Dsj), geo-touching (Tch), geo-inclusion (Inc), geo-crossing (CrS),

geo-pass-through (Pth), geo-overlapping (Ovl), geo-equality (Eql), geo-alias (Als) and geo-any (Any); and one metric operator: geo-distance (Dst).

The symbols used for the symbolic operators are  $\leftrightarrow$  for the geo-distance operator and  $\leftarrow$  "relation name"  $\rightarrow$  for the geo-any and geo-alias operators, (where "relation name" is *any* or *alias*).

Geo-any allows elimination of undesired constraints and, if a geo-any relationship is defined between a pair of SGOs, this means that no constraint exists between them. In contrast, geo-alias allows representation of more than one relationship (in OR) between a pair of SGOs. In fact, a graphical representation does not allow, for example, a pair of polygons that are both disjointed and overlapped.

One important issue is the definition of a sound method for analysis of syntactic and semantic correctness, of queries that may lead to multiple system and user interpretations. Thus, we illustrate an approach able to determine the exact syntactic and semantic interpretations of geographic configurations involved in queries expressed by a pictorial query language.

A visual query language with clear syntax and semantics can prevent a priori many ambiguities, minimizing multiple interpretations. The goal of this article is to present the configurations between geographical objects that can be considered syntactically correct in a geographical context, identify the set of GeoPQL operators referred to each configuration, and give each configuration a nonambiguous semantic.

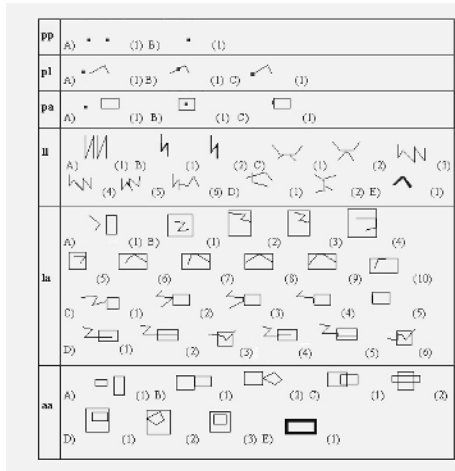
## **SYNTACTIC CORRECTNESS OF PICTORIAL CONFIGURATIONS**

In this section, a generic SGO pair, part of the set of all query SGOs, is considered. The set of operators syntactically admissible for this pair is defined, starting from possible spatial configurations (see Figure 2; Shekhar & Chawla, 2002). Thus the system considers for each configuration drawn by the user only the syntactically correct predicates, ensuring the syntactical correctness of the pictorial query. The proposed operators represent the constraints on the spatial properties of the objects (or classes) of the database that the user must specify in his or her query in order to find the geographic objects of interest.

Each possible spatial configuration is given a code of three alphabetic characters followed by a number. The first two characters indicate the type of SGO, the third indicates the type of spatial configuration, and the number distinguishes the configuration. In this manner, the first configuration between two polygons in Figure 2 is referred to as "aaA1," the last as "aaE1."



Figure 2. The different configurations considered



Let  $\psi_i$  and  $\psi_j$  be two SGOs that form a given configuration and are associated respectively with the database's geographical classes  $gc_\alpha$  and  $gc_\beta$ , possibly coincident.

For each configuration in Figure 2, a set of predicates  $\langle \psi_i \text{ Operator } \psi_j \rangle$  syntactically correct on the base of geometric and topological properties of  $\psi_i$  and  $\psi_j$  can be considered.

Table 1 summarizes syntactically correct predicates for all the configurations in Figure 2.

The operators geo-alias, geo-any, and geo-distance, in contrast to the other operators, must be expressed in the query using a suitable symbol (i.e., a labeled edge). Consequently, these operators are not linked to the particular configuration between the SGO pair involved, but to the symbols between them. For this reason, these kinds of operators require verification of the applicability requisites only, without considering the syntactical correctness of the configurations.

The geometric operators geo-union and geo-difference are applied only in relation to the specification of the properties (e.g., the value of the area) of their result (i.e., the object union or difference of the involved SGO). The property specification is sometimes related to the application of topological operators (e.g., the value of the intersection area between two SGOs).

## SEMANTIC CORRECTNESS OF PICTORIAL CONFIGURATIONS

In the previous table, all possible configurations of pairs of SGOs (operands) were considered. Their syntactical correctness was verified and the set of applicable operators identified. These properties are independent of the geographic database. In contrast with syntactical correct-

ness, semantic correctness is related to the meaning of the geographical information, so some syntactically correct configurations could be semantically incorrect, and some operators syntactically applicable to a configuration may be inapplicable semantically, due to the geographical information involved.

For example, the crossing of two polylines is always syntactically correct, and it is semantically correct, too, if the polylines represent two streets or a street and a river (the result could be a bridge), but it is semantically incorrect if they represent two rivers. It is obvious that semantic correctness is a subset of the syntactical correctness of the GeoPQL algebra operators applied to all possible pairs of SGOs.

Let  $\psi_i$  and  $\psi_j$  be two SGOs that form a given configuration and are associated respectively with the database's geographical classes  $gc_\alpha$  and  $gc_\beta$ , possibly coincident.

The predicate  $\langle \psi_i \text{ Operator } \psi_j \rangle$  is *semantically correct* if it is syntactically correct (for the configuration) and results in a non-null set of geographical objects. A configuration between the SGOs  $\psi_i$  and  $\psi_j$  is *semantically correct* if it is syntactically correct and at least one of its associated operators is semantically correct. Obviously, semantic correctness depends on the geographical classes  $gc_\alpha$  and  $gc_\beta$  associated with  $\psi_i$  and  $\psi_j$ .

For all the operators for which the symmetric property is not valid from the syntactical point of view, the semantic correctness depends on the order of the two operands. For example, a region can include a lake, but a lake cannot include a region.

If the result is a null set of objects, however, the user can autonomously define the predicate as "semantically correct, but absent in the database" for that configuration and manage "lists of semantic correctness for configurations, operators, and pair of geographical classes," or he or she can define the predicate as "semantically not correct" and manage "lists of semantic incorrectness for configurations, operators, and pair of geographical classes."

This procedure can be applied every time a new geographic class is defined. However, it can be very onerous because it is necessary to specify, for all configurations and operators, between the new class and previously defined classes giving a null result, whether they are correct or not.

Thus the system considers configurations and operators giving a non-null result as semantically correct. It considers all remaining configurations as undetermined until the user formulates a query that involves a  $\langle \psi_i \text{ Operator } \psi_j \rangle$  predicate for the pair of geographical classes. It is only then that the user decides if this construct is semantically correct, absent, or incorrect.

It is also important to specify the different role played, from the semantical point of view, by the topological

Table 1. Summary of syntactically correct operators for the different configurations considered

Configurations	Descriptions	Operators syntactically correct
ppA1	The configuration can represent two points $\psi_i$ and $\psi_j$ which are separate, or specify a property of their union.	$\psi_i$ Uni $\psi_j$ (or $\psi_j$ Uni $\psi_i$ ) $\psi_i$ Dsj $\psi_j$ (or $\psi_j$ Dsj $\psi_i$ )
ppB1	The points $\psi_i$ and $\psi_j$ are elements of different thematisms (classes) and their common property is their identical spatial coordinates	$\psi_i$ Eq1 $\psi_j$ (or $\psi_j$ Eq1 $\psi_i$ )
plA1	The configuration represents a point $\psi_i$ and a polyline $\psi_j$ which are separate.	$\psi_i$ Dsj $\psi_j$ (or $\psi_j$ Dsj $\psi_i$ )
plB1	In this configuration a point $\psi_i$ is located over a polyline and $\psi_j$ .	$\psi_j$ Tch $\psi_i$ $\psi_j$ Inc $\psi_i$
plC1	This is equal to the previous configuration and represents a point $\psi_i$ located over a polyline $\psi_j$ on the boundary.	$\psi_j$ Tch $\psi_i$ $\psi_j$ Inc $\psi_i$
paA1	The configuration represents two polygons $\psi_i$ and $\psi_j$ , which are separate.	$\psi_i$ Dsj $\psi_j$ (or $\psi_j$ Dsj $\psi_i$ )
paB1	The configuration represents a point $\psi_i$ located in a polygon $\psi_j$ .	$\psi_j$ Inc $\psi_i$
paC1	The configuration represents a point $\psi_i$ located on the boundary of the polygon $\psi_j$ .	$\psi_j$ Tch $\psi_i$ $\psi_j$ Inc $\psi_i$
IIA1	The configuration represents two polylines $\psi_i$ and $\psi_j$ , which are separate, or allow specification of a property of their union.	$\psi_i$ Dsj $\psi_j$ (or $\psi_j$ Dsj $\psi_i$ ) $\psi_i$ Uni $\psi_j$ (or $\psi_j$ Uni $\psi_i$ )
IIB1–IIB2	Two configurations can be considered: a) $\psi_j$ is completely within $\psi_i$ ; b) $\psi_j$ is completely within $\psi_i$ and they have one common boundary point.	$\psi_i$ Uni $\psi_j$ (or $\psi_j$ Uni $\psi_i$ ) $\psi_j$ Inc $\psi_i$ $\psi_j$ Dif $\psi_i$ $\psi_i$ Ovl $\psi_j$ (or $\psi_j$ Ovl $\psi_i$ )
IIC1–IIC2–IIC3–IIC4–IIC5–IIC6	These six configurations represent the different cases in which the two polylines $\psi_i$ and $\psi_j$ can have some, but not all, of their points in common, without crossing.	$\psi_i$ Tch $\psi_j$ (or $\psi_j$ Tch $\psi_i$ ) for IIC2, IIC3 and IIC6 $\psi_i$ Uni $\psi_j$ (or $\psi_j$ Uni $\psi_i$ ) always $\psi_j$ Dif $\psi_i$ for IIC1, IIC4, and IIC5 $\psi_i$ Ovl $\psi_j$ (or $\psi_j$ Ovl $\psi_i$ ) for IIC1, IIC4, and IIC5
IID1	These two configurations represent the different cases in which the two polylines $\psi_i$ and $\psi_j$ can have some, but not all, of their points in common, while crossing.	$\psi_i$ Uni $\psi_j$ (or $\psi_j$ Uni $\psi_i$ ) always $\psi_i$ Ovl $\psi_j$ (or $\psi_j$ Ovl $\psi_i$ ) for IID2 $\psi_i$ Crs $\psi_j$ (or $\psi_j$ Crs $\psi_i$ ) for IID1
IIE1	In this configuration the polylines $\psi_i$ and $\psi_j$ are elements of different thematisms (classes) and are spatially coincident.	$\psi_i$ Eq1 $\psi_j$ (or $\psi_j$ Eq1 $\psi_i$ )

operators in respect to the other operators. For example, if the query consists of two polylines that cross themselves, the syntactically correct predicates for this configuration are:

$$\begin{aligned} &\psi_i \text{ Crs } \psi_j \\ &\psi_i \text{ Uni } \psi_j \end{aligned}$$

The first predicate derives from a topological property of the configuration of the two SGOs involved in the query. Thus, it is semantically correct if the user explic-

itly specifies that the operator is semantically correct for that pair of operands, or if there is a set of geographical object pairs that satisfies this predicate in the database.

The second predicate is taken into consideration in the query only if the user specifies in it properties concerning the union of the SGOs. For this reason it is semantically correct if there are common properties between the two geographical classes  $gc_\alpha$  and  $gc_\beta$  represented by  $\psi_i$  and  $\psi_j$ .

Table 1. Summary of syntactically correct operators for the different configurations considered (cont.)

Configurations	Descriptions	Operators syntactically correct
laA1	The configuration represents a polyline $\psi_i$ and a polygon $\psi_j$ , which are separate.	$\psi_i$ Dsj $\psi_j$ (or $\psi_j$ Dsj $\psi_i$ )
laB1– laB2– laB3– laB4– laB5– laB6– laB7– laB8– laB9– laB10	These 10 configurations represent the various cases in which a polygon $\psi_j$ contains a polyline $\psi_i$ .	$\psi_i$ Inc $\psi_j$ always $\psi_i$ Tch $\psi_j$ (or $\psi_j$ Tch $\psi_i$ ) for laB2–laB3–laB4–laB5–laB6–laB7–laB8–laB9–laB10
laC1– laC2– laC3– laC4– laC5	These five configurations represent the various cases in which a polyline $\psi_i$ touches a polygon $\psi_j$ .	$\psi_i$ Tch $\psi_j$ (or $\psi_j$ Tch $\psi_i$ ) always $\psi_i$ Dif $\psi_j$ for laC3 and laC4 $\psi_j$ Inc $\psi_i$ for laC5
laD1– laD2– laD3– laD4– laD5– laD6	These six configurations represent the cases in which a polyline $\psi_i$ intersects a polygon $\psi_j$ .	$\psi_i$ Pst $\psi_j$ always $\psi_i$ Dif $\psi_j$ always $\psi_i$ Tch $\psi_j$ (or $\psi_j$ Tch $\psi_i$ ) for laD4–laD5–laD6
aaA1	The configuration represents two polygons $\psi_i$ and $\psi_j$ , which are separate, or allow a property of their union to be specified.	$\psi_i$ Dsj $\psi_j$ (or $\psi_j$ Dsj $\psi_i$ ) $\psi_i$ Uni $\psi_j$ (or $\psi_j$ Uni $\psi_i$ )
aaB1–aaB2	These configurations represent the cases in which two polygons $\psi_i$ and $\psi_j$ are touching.	$\psi_i$ Uni $\psi_j$ (or $\psi_j$ Uni $\psi_i$ ) $\psi_i$ Tch $\psi_j$ (or $\psi_j$ Tch $\psi_i$ )
aaC1– aaC2	These configurations represent the cases in which two polygons $\psi_i$ and $\psi_j$ overlap.	$\psi_i$ Uni $\psi_j$ (or $\psi_j$ Uni $\psi_i$ ) $\psi_i$ Dif $\psi_j$ (or $\psi_j$ Dif $\psi_i$ ) $\psi_i$ Ovl $\psi_j$ (or $\psi_j$ Ovl $\psi_i$ )
aaD1– aaD2– aaD3	These configurations represent the cases in which the polygon $\psi_i$ encloses the polygon $\psi_j$ .	$\psi_i$ Uni $\psi_j$ (or $\psi_j$ Uni $\psi_i$ ) always $\psi_i$ Dif $\psi_j$ always $\psi_i$ Inc $\psi_j$ always $\psi_i$ Ovl $\psi_j$ (or $\psi_j$ Ovl $\psi_i$ ) always $\psi_i$ Tch $\psi_j$ (or $\psi_j$ Tch $\psi_i$ ) for aaD1 and aaD2
aaE1	In this configuration, the polygons $\psi_i$ and $\psi_j$ are elements of different thematisms (classes) and are spatially coincident.	$\psi_i$ Eql $\psi_j$ (or $\psi_j$ Eql $\psi_i$ )

## FUTURE TRENDS

A query on a geographical database involves the following two types of information properties:

- *spatial properties* (e.g., the topological relationships or the distance). These properties are verified by the geometric attributes of different objects which form the database; and
- *descriptive properties* (e.g., the population of a given region, or the length of a given river). These properties are verified by the alphanumeric attributes of the aforementioned objects.

Future trends in query languages for GIS refer to the representation of spatial properties using visual approaches. However, if the use of visual approaches allows to the user to specify, in a graphical way, topological and geometric properties among all the objects involved in the

query, some problem related to their use could arise. Other works refer to studies regarding the meaningful power, completeness, and unambiguousness of visual query languages.

## CONCLUSION

The use of nonprocedural visual query languages allows the user to query geographical databases almost without knowing procedures and syntax of these languages. In particular, by pictorial query languages, user friendliness and ambiguous queries is no longer an alternation. Different problems are still open and are subjects of studies, as is the advancement the technology brings to new proposals and solutions for these kinds of languages.

## REFERENCES

Aufaures-Portier, M. A. (1995). A high level interface language for GIS. *Journal of Visual Languages and Computing*, 6(2), 167-182.

Aufaures-Portier, M. A., & Bonhomme, C. (1999). A high level language for spatial data management. *Proceedings on Visual Information Systems (VISUAL '99)*, 1614 (pp. 325-332). Amsterdam, The Netherlands: Springer-Verlag.

Blaser, A. D., & Egenhofer, M. J. (2000). A visual tool for querying geographic databases. *Proceedings on Advances in Visual Interfaces* (pp. 211-216). Palermo, Italy: ACM.

Calcinelli D., & Mainguenaud, M. (1994). Cigales, a visual language for geographic information system: The user interface. *Journal of Visual Languages and Computing*, 5(2), 113-132.

Chrisman, N. (2002). *Exploring geographic information systems*. Wiley.

Egenhofer, M. J. (1997). Query processing in spatial-query-by-sketch. *Journal of Visual Languages and Computing*, 8(4), 403-424.

Favetta, F., & Aufaure-Portier, M. A. (2000). About ambiguities in visual GIS query languages: A taxonomy and solutions. *Proceedings on Visual Information, 1929* (pp. 154-165). Lyon, France: Springer-Verlag.

Ferri, F., Massari, F., & Rafanelli, M. (1999). A pictorial query language for geographic features in an object-oriented environment. *Journal of Visual Languages and Computing*, 10(6), 641-671. Academic Press.

Ferri, F., & Rafanelli, M. (2004). *GeoPQL: A geographical pictorial query language* (Tech. Rep). IASI-CNR.

Laurini, R., & Thompson, D. (1992) *Fundamentals of spatial information systems*. Academic Press.

Mainguenaud, M., & Portier, M. A. (1990). Definition of Cigales: A GIS query language. *Proceedings on Databases and Expert Systems Applications* (pp. 275-280). Vienna, Austria: Springer-Verlag.

Meyer, B. (1993). Beyond icons: Towards new metaphors for visual query languages for spatial information systems. *Proceedings of the 1<sup>o</sup> International Workshop on Interfaces in Database Systems* (pp. 113-135). Glasgow, UK: Springer-Verlag.

Rigaux, P., Scholl, M. O., & Voisard, A. (2001). *Spatial databases: With application to GIS*. Morgan Kaufmann.

Shekhar, S., Chawla, S., Ravada, S., Fetterer, A., Liu, X., & Lu, C. (1999) Spatial databases: Accomplishments and research needs. *IEEE Transactions on Knowledge and Data Engineering*, 10(1), 45-55.

Shekhar, S., & Chawla, S. (2002). *Spatial databases: A tour*. Prentice Hall.

## KEY TERMS

**Blackboard Metaphor:** A metaphor used in query languages, its philosophy is to let users draw the sketch of their query.

**Declarative (Nonprocedural) Query Language:** A general term for a query language, as opposed to an imperative query language. Imperative (or procedural) languages specify explicit sequences of steps to follow to produce a result, while declarative (nonprocedural) languages describe relationships between variables in terms of functions or inference rules, and the language executor (interpreter or compiler) applies some fixed algorithm to these relations to produce a result.

**Geographical Database:** A database in which geographical information is store by  $x$ - $y$  coordinates of single points, or points that identify the boundaries of lines (or polylines, which sometimes represent the boundaries of polygons). Different attributes characterize the objects stored in these databases. In general, the storing structure consists of "classes" of objects, each of them implemented by a layer. Often a geographical database includes raster, topological vector, image processing, and graphics production functionality.

**Geographical Information System (GIS):** A computerized database system used for the capture, conversion, storages, retrieval, analysis, and display of spatial objects.

**Metaphor:** Figurative language that creates an analogy between two unlike things. Metaphor does not make a comparison but creates its analogy by representing one thing as something else.

**Pictorial Query Language:** A general term for a query language, as opposed to a textual query language. Pictorial languages describe the result to produce characterized by, or composed of, pictures.

**Visual Query Language:** A language that allows the user to specify its goals in a two- (or more) dimensional way with visual expressions—spatial arrangements of textual and graphical symbols.

# Temporal Databases

**Mahesh S. Raisinghani**

*Texas Woman's University, USA*

**Chris Klassen**

*University of Dallas, USA*

## INTRODUCTION

Information has emerged as an agent of integration and the enabler of new competitiveness for today's enterprise in the global marketplace. The degree of change in the paradigm for storage of data in databases is examined to determine whether it can support the accelerated response time required for information systems and technology. This paper discusses the key concepts for understanding temporal databases, including major data types and its principal purpose. Moreover, once the temporal extension is added to existing ANSI and ISO SQL standards, it will enable users to take advantage of new temporal features in the major database products.

Time has always been of great interest to mankind. In recent years, the computer industry has made a great influence in human lifestyle and it like other industries is time prone. For instance, the year 2000 problem (Y2K) was time-related. Not surprisingly, researchers working in the field of databases are expressing increasing interest in the dimension of time. Classical database design is two-dimensional and contains only current data, which can be termed snapshot data type. Today's businesses must adapt constantly to an ever-changing business environment, and databases must support an evolving business framework. More and more time-management seminars and devices are introduced everyday. As Snodgrass (1998) notes, "Time varying data is becoming pervasive. It has been estimated that one of every 50 lines of database application code involves a date or time value."

## BACKGROUND: THE TIME DIMENSION IN DATABASES

The classical database is two-dimensional with columns and rows that intersect each other at cells, which contain particular values. Now extend this flat two-dimensional (2D) database into a three-dimensional (3D) figure, such as a cube. Apply this 3D concept to a database,

instead of a flat 2D construct, and now you have an extended 3D figure with the third dimension being represented as various time intervals (Tansel et al., 1993). Date, Darwen, and Lorentzos (2003) provide a detailed investigation into the application of interval and relational theory to the problem of temporal database management.

Jensen, Mark, Roussopoulos, and Sellis (1993) presented an architecture for query processing in the relational model extended with transaction time that integrates standard query optimization and computation techniques with new differential computation techniques. The use of differential computation techniques is essential in order to provide efficient processing of queries that access very large temporal relations.

Temporal database systems are systems that provide special support for storing, querying, and updating historical and future data (Date et al., 2003). A temporal database records time-variant information. Date (2004) states that the relational model needs no extension or corruption in order to support the time dimension. Snodgrass (1998) defines the temporal database as "a database that supports some aspect of time". Another definition that may be better structured to fit a Temporal Relational Model states that a temporal database is defined as "an union of two sets of relations  $R_s$  and  $R_t$ , where  $R_s$  is the set of all static relations and  $R_t$  is the set of all time-varying relations" (Navathe & Ahmed, 1993). This article is limited to the consideration of the relational model of the temporal database to the exclusion of the other well-known types of databases such as object-oriented, network, and hierarchical. While there is little temporal database research currently on the latter two types of databases, an increasing amount of research is being done in the area of temporal object-oriented databases. We might also note that temporal databases have also been referred to as time-oriented databases, time-varying databases, or historical databases. While time-oriented database and time-varying database are equivalent in meaning to temporal database, historical database is not. As discussed later in this article, a historical database is actually a subset of the temporal database.



The first thing to note about the Snodgrass (1998) definition of a temporal database is that the time dimension requirement for a database to be temporal does not include user-defined time. User-defined time is some aspect of time that is not recognized by the database management system as a special data type. A classical database essentially treats data, such as birthdates, as text strings. This treatment of date data does not allow for much manipulation. One way of attempting to avoid this problem is to treat the date as a type of text data to store the date as a number. This approach, however, has its own problem set. One such problem found in Oracle version 7.2 is that in some instances in year 2000, dates are treated as earlier than dates in the 1900s. Converting dates to numbers internally may be best way of making possible a wide range of ad hoc queries on temporal data. The fact that support of user-defined time does not merit a database being considered temporal and does not mean that the temporal database cannot or will not include user-defined time.

## TEMPORAL DATABASE CHARACTERISTICS

There are several types of time in a temporal database. Before describing these different types in detail, it is important to clarify the concept of precision relating to the time stored in a temporal database. This concept denoting precision in a temporal database is called the granularity of time (Howe, 1997). An example of progressive time granularity includes a day, an hour, a second, or a nanosecond.

It is important to understand three major concepts to grasp the precise nature of a temporal database. The first is the concept of valid time. The second concept is transaction time, the actual time recorded in the database at the time the data is entered. Time stamps can include either the date or the date and clock time. An object is an entity that has a well-defined role in the application domain, and its features include state, behavior, and identity. An employee is a good example of an object. In a classical database, once a change is made to an employee's record, original data is changed, discarded, and replaced by new data. However, in a temporal database, which supports transaction time, transaction time can be attached in the form of a time stamp to both the old data and to the new data for that employee. In so doing, the database can store both the old data and new data for the same object. In this case, the salary of the employee was increased on a certain date. It is important to note here that the transaction time values or time stamps cannot be later than the current point in

time nor can they be changed, just as the past cannot be changed.

Another major type of time in a temporal database is termed as valid time. Valid time is the actual or real-world clock time at which the data is valid. Continuing with the employee example, while transaction time is the point in time at which the data is entered into the database, valid time is the unique point when the entered data become true or take effect. For instance, on January 3, an employee is notified that an increase in salary will be effective February 1. The Human Resources department, after being notified of the employee's raise, must enter the new salary into the database. Presumably, they will enter the data before the raise goes into effect. The actual time they enter the raise into the database is prior to February 1st, and that time will be the time stamp for the transaction time. The data, however, are not yet valid, and for the rest of January, the employee will continue to receive the current salary. However, on February 1, the raise data will become valid. Thus, February 1 is the valid time. Also, if an employee receives a raise that is retroactive, the transaction time may be later than the valid time.

The two major types of time unique to the temporal database, valid time and transaction time, allow for the possibility of three forms of temporal databases: historical, rollback, and bi-temporal (Steiner, 2003). A historical database supports valid time, but not transaction time. To reiterate, a historical database is a poor choice as an alternative term for the temporal database. The reason is that a historical database is but one type of temporal database. A historical database, however, as explained later, would be a poor choice for anyone wishing to deploy a temporal database. The second form of a temporal database is the rollback database. This database is the opposite of the historical database. The rollback database only supports transaction time and not valid time. As opposed to the historical database, rollback database is quite useful for data recovery after database failure. The reason then that a temporal database would rarely be desirable is that it could not support rollback after DBMS failure. It is also necessary if the database does not use the locking technique to ensure data security. Most databases on the market today do support at least some rollback features.

In reality, a temporal database is a bi-temporal database. This database supports both types of time that are necessary for storing and querying time-varying data. The bi-temporal database could aid significantly in knowledge discovery, since it is able to fully support the time dimension on three levels: the DBMS level with transaction time, the data level with valid time, and the user-level with user-defined time.

## DATATYPES IN A TEMPORAL DATABASE

A temporal database supports three major data types: temporal, static, and snapshot data. It is important that the data types be transparent since the user need not know whether the data used is temporal, static, or snapshot. One goal of temporal database should be provision of seamless handling of temporal, static, and snapshot data (Gadia & Nair, 1993).

The temporal datatype is the most important datatype for the temporal database, for the simple fact that it provides the foundation for building the temporal database. The temporal datatype has been defined as “a finite union of intervals” (Gadia & Nair, 1993). This datatype is termed *temporal element*. The temporal element is necessary for using time-varying criteria to perform ad-hoc queries. Later in this article, an example of a temporal querying language is presented, which uses the temporal element in a new temporal clause that is added to the existing SQL syntax.

Static datatype is defined as “a constant defined over the whole universe of time” (Gadia & Nair, 1993). In other words, the validity of a static value is defined as any future time. In contrast, temporal datatype value is valid for a specified time period or interval, and a snapshot datatype value is valid only for the current instant. Another way of expressing the validity of static data is to say that its domain is any finite point in the future. Without a finite period of validity, the static datatype does not need a time stamp.

The final major datatype is termed *snapshot* data. In a conventional or classical database, all data is of the snapshot datatype (Gadia & Nair, 1993; Navathe & Ahmed, 1993; Ozsoyoglu & Snodgrass, 1995). When tables are updated, the new values replace old values, which are discarded, and a new snapshot is created. Unfortunately, with the snapshot datatype, the history of the changes to the data is lost. Thus, the database should support the history of the data for a given business, so merely having a snapshot datatype supported by a database is unacceptable, hence, proving the need for temporal database. Another reason to retain the capability for the snapshot datatype is to provide a smooth transition from a classical database to a temporal database. Since all the data in a classical database is of the snapshot datatype, migration from a classical database to a temporal database should be seamless (Gadia & Nair, 1993).

## QUERY CAPABILITIES FOR THE TEMPORAL DATABASE

One of the most important reasons for having a database that supports the temporal dimension is the ability to perform ad-hoc queries about the data. The current standard for conventional (relational) databases is Structured Query Language (SQL). SQL has become the industry standard for Relational Database Management Systems (RDBMSs) because of its ease of use due to its English-like syntax. The addition of the temporal dimension, however, greatly increases the complexity of the queries using temporal data. With the additional element of time, SQL in its current form is no longer able to process ad-hoc queries as was done formally using a (relational) classical database. A new query language or extension to SQL is necessary. Understandably, one of the biggest areas of research today in the field of temporal databases is focused on this topic. However, a few remarks on this topic are provided, and one example of a query language extension is given in the following discussion.

Any new temporal query language should support current SQL capabilities seamlessly. The most promising proposals for a query language which supports the time dimension and also continues to allow users to perform queries without specifying any time criteria (thus avoiding high retraining costs, among other things) are not really new query languages, per se, but are simply extensions of SQL. One of the most promising examples of temporal-capable SQL is called, appropriately, TSQL for Temporal Structured Query Language. With TSQL, ad-hoc queries can be performed without specifying any time-varying criteria. Thus, the only essential clauses that are necessary to perform a query in TSQL remain the same as in SQL. These are the SELECT and FROM clauses. If desired, a new criteria clause can be added to current SQL criteria clauses, which are WHERE, GROUP BY, HAVING, and ORDER BY. The new criteria clause would be either WHEN or WHILE. Either name captures the idea of a time-varying condition. Böhlen, Jensen, and Snodgrass (2000) advocate a different approach to articulating a set of requirements that directly implies the syntactic structure and core semantics of a temporal extension of an (arbitrary) nontemporal query language. This extended language, termed ATSQL, is formally defined via a denotational-semantics-style mapping of tempo-

ral statements to expressions using a combination of temporal and conventional relational algebraic operators. Currently, a proposal to add TSQL to the existing ANSI and ISO SQL standards is under consideration by the respective governing bodies.

## BUSINESS APPLICATIONS

One of the aims of this article is the integration of the theoretical and practical aspects of temporal databases. To this point, the presentation has dealt with theoretical aspects of temporal databases. Now, we will look at some of the business applications of the temporal databases. The first type is the data warehouse. Data warehouses are enterprise-wide databases. These are enormous corporate databases, which are generally used to store all information about a company. As stated earlier, the element of time is crucial to many businesses, and it is in the data warehouse that the time dimension can be stored and utilized for querying the data. It is important to relate data warehouses with temporal databases. Every data warehouse has a time dimension that makes every data warehouse a temporal database (Kimball, 1997; Snodgrass, 1998). “The time dimension is a unique and powerful dimension in every datamart and enterprise data warehouse” (Kimball, 1997). Golan and Edwards (1993) discuss how temporal rules for trading at the Toronto Stock Exchange (TSE) were developed by First Marathon Securities in Canada, who derived relationships between 32 stock and economic indicators including TSE, DOW, S&P, P/E, and seven major industry indexes over a six-month period. The rules describe stock behavior of major TSE trading companies and their sensitivity to market or economic indices. The reported benefits include the description of relationships and stocks’ sensitivities to various indices.

Stantic, Khanna, and Thornton (2004) state that most modern database applications contain a significant amount of time-dependent data and a substantial proportion of this data is *now-relative*, that is, current *now*. Their research addresses the indexing of now-relative data, which is a natural and meaningful part of every temporal database as well as the focus of most queries. They propose a logical query transformation that relies on the representation of current time and the geometrical features of spatial access methods, and empirically demonstrate that this method is efficient on now-relative bi-temporal data, outperforming a straightforward maximum time stamp approach by a factor of more than 20, both in number of disk accesses and CPU usage.

The scientific laboratory is another good business application where temporal database plays an important

role. A laboratory performs many tests and many versions of the same test. Each test often involves precise timing sequences. To ensure that time information is not lost, it must be stored in a database. Since the element of time in these tests cannot be discarded, the database in which the information is stored must be a temporal database. One of the major benefits of storing scientific test information in a temporal database that supports a temporal query language is that new patterns and knowledge can be mined from the database (Loomis, 1997). The third type of business application relies heavily on sales and marketing data. Success in sales and marketing depends heavily on effective timing. The appropriate time to approach a customer and when to advertise in certain locations is the knowledge that is essential for such types of businesses. The ability to correctly forecast this information is best provided by a temporal database. Some of the uses for temporal databases are in forecasting future events and analyzing patterns.

The last type of business application is multimedia. Multimedia is one of the fastest growing areas of business today, especially as conducted over the Internet. The time will soon come when a person will be able to order movies over the Internet and view them on a personal computer. The process is as follows: the person orders the movie, video images are stored in a multimedia database, and these are synchronized with the audio files in either the same or a different database. Synchronization means simultaneity and involves a time element.

Because multimedia is time-based, a great deal of synchronization is necessary. These media components will display in a precise synchronized order. Multimedia objects have temporal and spatial relationships that must be maintained in their presentation. Some of this data has real-time constraints in its delivery to client stations (Ozsu, 2003).

Thus, the multimedia database is another instance of a temporal database application like data warehousing.

## THE FUTURE OF TEMPORAL DATABASES

Since the foundations and principles of temporal databases are firmly rooted in the relational model of data and represent an evolutionary step, not a revolutionary one, they will stand the test of time (Date et al., 2003). As the business environment becomes increasingly competitive, the knowledge within an organization becomes an invaluable resource. To track this information, businesses must implement temporal databases. To provide

businesses with this capability to keep track of time-varying data, two developments are foreseen. Tuzhilin (1993) provided a good example of the benefits of integrating a temporal database with an existing business system to increase knowledge.

Second, a temporal extension will be added to existing ANSI and ISO SQL standards. This development will enable users to take advantage of new temporal features in the major database products. This extension will allow better quality and quantity of knowledge to be stored in and mined from the newly temporalized databases and gain competitive advantage. Mokbel, Xiong, and Aref (2004) discuss the Scalable Incremental hash-based Algorithm (SINA), a new algorithm for evaluating a set of concurrent continuous spatio-temporal queries. This algorithm was designed to incrementally evaluate continuous spatio-temporal queries and provide scalability in terms of the number of concurrent continuous spatio-temporal queries. The experimental results show that SINA is scalable and is more efficient than other index-based spatio-temporal algorithms.

## CONCLUSION

With respect to the expected role and impact of the time construct and its usages in future database systems, active mechanisms based on event-condition-action rules will play an important role in next-generation database management systems. This can be useful to several novel applications including planning, scheduling, project management, medical information systems, geographical information systems, and natural language processing systems. Finally, although many temporal extensions of the relational data model have been proposed, there is no comparable amount of work in the context of object-oriented data models and extend the standard for object-oriented databases, defined by Object Data Management Group (ODMG) using temporal extensions. The investigation, on a formal basis, of the main issues arising from the introduction of time in an object-oriented model could lead to developments in spatio-temporal databases supporting spatial objects with continuously changing position and extent. The graphical user interfaces for spatio-temporal information, query processing in spatio-temporal databases, and storage structures and indexing techniques for spatio-temporal databases could have a strategic impact on business applications such as data mining and knowledge discovery.

## REFERENCES

- Böhlen, M.H., Jensen, C.S., & Snodgrass, R.T. (2000, December). Temporal statement modifiers. *ACM Transactions on Database Systems*, 25(4), 407-456.
- Date, C.J. (2003). On modeling history. Retrieved January 29, 2005, from <http://www.dbdebunk.com/page/page/1027290.htm>
- Date, C.J. (2004). *Introduction to database systems* (8th ed.). Addison-Wesley.
- Date, C.J., Darwen, H., & Lorentzos, N.A. (2003). *Temporal data and the relational model*. Morgan Kaufmann.
- David, M.M. Multimedia databases through the looking glass. Database Programming & Design Online. Retrieved January, 29, 2005, from <http://www.dbpd.com/9705david.htm>
- Gadia, S.K., & Nair, S.S. (1993). Temporal databases: A prelude to parametric data. In A. Tansel et al. (Eds.), *Temporal databases: Theory, design, and implementation* (pp. 28-66). Benjamin/Cummings.
- Golan, R., & Edwards, D. (1993, July). Temporal rules discovered using DataLogic/R with stock market data. *RSKD-93*, 61-72.
- Jensen, C.S., Mark, L., Roussopoulos, N., & Sellis, T. (1993, January). Using differential techniques to efficiently support transaction time. *The International Journal on Very Large Data Bases*, 2(1), 75-116.
- Jensen, C.S., et al. (1993). Glossary. In A. Tansel et al. (Eds.), *Temporal databases: Theory, design, and implementation* (pp. 621-633). Benjamin/Cummings.
- Kimball, R. (1997, July). It's time for time. DBMS Online. Retrieved January 29, 2005, from <http://www.dbmsmag.com/9707d05.html>
- Loomis, T. (1997, August). Audit history and time-slice archiving in an object DBMS for laboratory databases. *Hewlett Packard Journal*.
- Mokbel, M.F., Xiong, X., & Aref, W.G. (2004). SINA: Scalable incremental processing of continuous queries in spatio-temporal databases. *ACM Press*, 623-634.
- Navathe, S.B., & Rafi, A. (1993). Temporal extensions to the relational model and SQL. In A. Tansel et al. (Eds.), *Temporal databases: Theory, design, and implementation* (pp. 92-109). Benjamin/Cummings.



Ozsoyoglu, G., & Snodgrass, R.T. (1995, August). Temporal and real-time databases: A survey. *IEEE Transactions on Knowledge and Data Engineering*.

Ozsu, M.T. (0000). A new foundation. Database Programming & Design Online. Retrieved January 29, 2005, from <http://www.dbpd.com/ozsu.htm>

Snodgrass, R.T. (1998, June). Of duplicates and septuplets. *Database Programming & Design*, 46-49.

Stantic, B., Khanna, S., & Thornton, J. (2004). An efficient method for indexing now-relative bitemporal data. *ACM International Conference Proceeding Series*, 27, 113-122.

Steiner, A. (2003). What are temporal databases? Retrieved January 29, 2005, from <http://www.inf.ethz.ch/personal/steiner/TemporalDatabases/TemporalDB.html>

Tansel, A., et al. (Eds.). (1993). Part I: Extensions to the relational model. *Temporal databases: Theory, Design, and implementation* (pp. 1-5). Benjamin/Cummings.

Tuzhilin, A. (1993). Applications of temporal databases to knowledge-based simulations. In A. Tansel et al. (Eds.), *Temporal databases: Theory, design, and implementation* (pp. 580-593). Benjamin/Cummings.

## KEY TERMS

**Bi-Temporal Database:** This database supports both types of time that are necessary for storing and querying time-varying data. It aids significantly in knowledge discovery, because only the bi-temporal database is able to fully support the time dimension on three levels: the DBMS level with transaction time, the data level with valid time, and the user-level with user-defined time.

**Granularity of Time:** A concept that denotes precision in a temporal database. An example of progressive time granularity includes a day, an hour, a second, or a nanosecond.

**ODMG:** Object Data Management Group. The ODMG group completed its work on object data management standards in 2001 and was disbanded. The ODMG submitted the ODMG Java Binding to the Java Community Process as a basis for the Java Data Objects Specification (JDO). JDO provides transparent persistence (<http://www.odmg.org/>).

**RDBMS:** Relational Database Management Systems.

**Spatio-Temporal Database:** A spatiotemporal element is a finite union of spatio-temporal intervals. Spatio-temporal elements are closed under the set theoretic operations of union, intersection, and complementation. Databases that support spatial objects with continuously changing position and extent. The graphical user interfaces for spatio-temporal information, query processing in spatio-temporal databases, and storage structures and indexing techniques for spatio-temporal databases could have a strategic impact on business applications such as data mining and knowledge discovery.

**SQL:** Structured Query Language is a standard interactive and programming language for getting information from and updating a database. Although SQL is both an ANSI and an ISO standard, many database products support SQL with proprietary extensions to the standard language. Queries take the form of a command language that lets you select, insert, update, find out the location of data, and so forth. There is also a programming interface.

**Temporal Database:** A temporal database is a database that records time-variant information. It is defined as an union of two sets of relations  $R_s$  and  $R_t$ , where  $R_s$  is the set of all static relations and  $R_t$  is the set of all time-varying relations.

**TSQL:** Temporal Structured Query Language. With TSQL, ad hoc queries can be performed without specifying any time-varying criteria. The new criteria clause would be either 'WHEN' or 'WHILE'. Either name captures the idea of a time-varying condition. Currently, a proposal to add TSQL to the existing ANSI and ISO SQL standards is under consideration by the respective governing bodies.



# Text Categorization

**Fabrizio Sebastiani**

*Università di Padova, Italy*

## INTRODUCTION

During the last 15 years, the production of documents in digital form has exploded, due to the increased availability of hardware and software tools for generating digital data (e.g., personal computers, digital cameras, word processors) and for digitizing data that had been originated in nondigital form (e.g., scanners, OCR software). This phenomenon has also strongly affected “novel” digital media such as imagery, video, music, and so forth. However, natural language text has been, at least from a quantitative viewpoint, the medium most responsible for this explosion, due to its immediacy and to the ubiquity of word processing and text authoring tools. As a consequence, there is an increased need for hardware and software solutions for storing, organizing, and retrieving the large amounts of digital text that are being produced, with an eye towards its future use.

The design of such solutions has traditionally been the object of study of *information retrieval* (IR), the discipline that is broadly concerned with the computer-mediated access to data with poorly specified semantics. While all of the previously mentioned types of media fall within the scope of IR, it is unquestionable that text has been its major focus of attention ever since its inception in the late 1950s.

The following are two main directions one may take for providing convenient access to a large, unstructured repository of text:

- **Providing powerful tools for searching relevant documents within this large repository.** This is the aim of *text search*, a subdiscipline of IR concerned with building systems that accept a natural language query and return as a result a list of documents ranked according to their estimated relevance to the user’s information need. Nowadays, the “tip of the iceberg” of text search is represented by Web search engines, but commercial solutions for the text search problem were being delivered decades before the birth of the Web.
- **Providing powerful tools for turning this unstructured repository into a structured one, thereby easing storage, search, and browsing.** This is the aim of *text classification*, a subdiscipline of IR concerned with building systems that partition an

unstructured collection of documents into meaningful groups.

There are two main variants of text classification. The first is *text clustering*, which is concerned with finding a latent yet undetected group structure in the repository, and the second is *text categorization* (TC), which is concerned with structuring the repository according to a scheme given as input. In other words, while in the former task the set of groups (or classes, or labels) is not known in advance, it is predefined and known in the latter. The latter task will be the focus of this paper.

Note that the underlying notion of TC, that of membership of a document  $d_j$  in a class  $c_i$  (based on the semantics of  $d_j$  and  $c_i$ ), is inherently subjective. This is because different classifiers (be they human or machine) might disagree on whether  $d_j$  belongs in  $c_i$ . This means that membership cannot be determined with certainty, which in turn means that any classifier (be it human or machine) will be prone to misclassification errors. It is thus customary to evaluate text classifiers by applying them to a set of labelled (i.e., preclassified) documents (which here plays the role of a “gold standard”). In this way, the *accuracy* of the classifier may be measured by the degree of coincidence between its classification decisions and the labels originally attached to the documents.

## Applications

Maron’s (1961) seminal paper is usually taken to mark the official birth date of TC, which at the time was called *automatic indexing*. This name reflected the fact that the main (or only) application that was then envisaged for TC was to automatically index (i.e., generating internal representations for) scientific articles for Boolean information retrieval systems. In fact, since index terms for these representations were drawn from a fixed, predefined set of such terms, we can regard this type of indexing as an instance of TC once index terms play the role of classes. The importance of TC increased in the late ‘80s and early ‘90s with the need to organize the increasingly larger quantities of digital text being handled in organizations at all levels. Since then, frequently pursued applications of TC technology have been

- *newswire filtering* (i.e., the grouping of news stories produced by news agencies according to thematic classes of interest; Hayes & Weinstein, 1990);
- *patent classification* (i.e., the organization of patents into taxonomies so as to ease the detection of existing patents related to a new patent; Fall, Törösvári, Benzineb, & Karetka, 2003); and
- *Web page classification* (i.e., the grouping of Web pages [or sites] according to the taxonomic classification schemes typical of Web portals; Dumais & Chen, 2000).

The previous applications all have a certain thematic flavour, in the sense that classes tend to coincide with topics, or disciplines. However, TC technology has been applied to domains that are not thematic in nature, among which are

- *spam filtering* (i.e., the grouping of personal e-mail messages into the two classes [LEGITIMATE and SPAM] so as to provide effective user shields against unsolicited bulk mailings; Drucker, Vapnik, & Wu, 1999);
- *authorship attribution* (i.e., the automatic identification of the author of a text among a predefined set of candidate authors (Diederich, Kindermann, Leopold, & Paaß, 2003);
- *author gender detection* (i.e., a special case of the previous task in which the issue is deciding whether the author of the text is a MALE or a FEMALE; Koppel, Argamon, & Shimoni, 2002);
- *genre classification* (i.e., the identification of the nontopical nature of the text, such as determining if a product description is a PRODUCT REVIEW or an ADVERTISEMENT; Stamatatos, Fakotakis, & Kokkinakis, 2000);
- *survey coding* (i.e., the classification of respondents to a survey based on the textual answers they have returned to an open-ended question; Giorgetti & Sebastiani, 2003); or even
- *polarity detection* (i.e., deciding if a product review is THUMBS UP or a THUMBS DOWN; Pang, Lee, & Vaithyanathan, 2002).

## TECHNIQUES

### Approaches

In the 1980s, the most popular approach to TC was one based on knowledge engineering, whereby a knowledge engineer and a domain expert working together built an expert system that automatically classified text. Typically, such an expert system would consist of a set of “if ... then

...” rules, to the effect that a document was assigned to the class specified in the “then” clause only if the linguistic expressions (i.e., words) specified in the “if” part occurred in the document. The drawback of this approach was the high cost of humanpower required for defining the rule set and maintaining it (i.e., for updating the rule set as a result of possible subsequent additions or deletions of classes or as a result of shifts in the meaning of the existing classes).

In the 1990s, this approach was superseded by the supervised machine learning approach, whereby a general inductive process (the *learner*) is fed with a set of “training” documents, preclassified according to the categories of interest. By observing the characteristics of the training documents, the learner may generate a model (the *classifier*) of the conditions that are necessary for a document to belong to any of the categories considered. This model can subsequently be applied to previously unseen documents for classifying them according to these categories.

This approach has several advantages over the knowledge engineering approach. First of all, a higher degree of automation is introduced: The engineer needs to build not a text classifier, but an automatic builder of text classifiers (the learner). Once built, the learner can then be applied to generating many different classifiers for many different domains and applications; one only needs to feed it with the appropriate sets of training documents. By the same token, the previously mentioned problem of maintaining a classifier is solved by feeding new training documents appropriate for the revised set of classes. Many inductive learners are available off the shelf; if one of these is used, the only humanpower needed in setting up a TC system is that for manually classifying the documents to be used for training. For performing this latter task, less skilled humanpower is needed than for building an expert system, which is also advantageous. Consider also that, when an organization has previously relied on manual work for classifying documents, many preclassified documents are already available to be used as training documents when the organization decides to automate the process.

Most important, the accuracy of classifiers (i.e., their capability to make the right classification decisions) built by machine learning methods now rivals that of human professionals and usually exceeds that of classifiers built by knowledge engineering methods. This has brought about a wider acceptance of supervised learning methods, even outside academia. Although for certain applications (such as spam filtering) a combination of machine learning and knowledge engineering is still the basis of several commercial systems, it is fair to say that in most other TC applications (especially of the thematic type), the adoption of machine learning technology has been widespread.

### Learning Text Classifiers

Many different types of supervised learners have been used in TC (Sebastiani, 2002), including probabilistic “naive Bayesian” methods, Bayesian networks, regression methods, decision trees, Boolean decision rules, neural networks, incremental or batch methods for learning linear classifiers, example-based methods, classifier ensembles (including boosting methods), and support vector machines. The time span between the development of a new, supervised learning method and its application to TC has become narrower because machine learning researchers now view TC as a strategic and challenging application and one of the benchmarks of choice for the algorithms they develop. Although all of the techniques mentioned previously still retain their popularity, it is fair to say that in recent years support vector machines (Joachims, 1998) and boosting (Schapire & Singer, 2000) have been the two dominant learning methods in TC. This seems due to a combination of two factors: (a) these two methods have strong justifications in terms of computational learning theory, and (b) in comparative experiments on widely accepted benchmarks, they have outperformed all other competing approaches.

### Building Internal Representations for Documents

The learners discussed in the previous section cannot operate on the documents as they are; the documents must be given internal representations that the learners (and classifiers, once built) can make sense of. It is thus customary to transform all the documents (i.e., those used in the training phase, in the testing phase, or in the operational phase of the classifier) into internal representations by means of methods used in text search, where the same need is also present. By means of these methods, a document is usually represented by a vector, where the dimensions of the vector correspond to the terms that occur in the training set, and the value of each individual entry corresponds to the weight that the term in question has for the document.

In TC applications of a thematic kind, the set of terms is usually made to coincide with the set of content-bearing words (i.e., all words but topic-neutral words such as articles, prepositions, etc.), possibly reduced to their morphological roots (stems) so as to avoid excessive stochastic dependence among different dimensions of the vector. Weights for these words are meant to reflect the word’s importance in determining the semantics of the document in which it occurs and are automatically computed by weighting functions. These functions usually rely on intuitions of a statistical kind, such as

- the more often a term occurs in a document, the more important it is for that document; and
- the more documents a term appears in, the less important that term is in characterizing the semantics of those documents.

In nonthematic TC applications, the opposite is often true. For instance, frequently used articles, prepositions, and punctuation (together with many other stylistic features) may be helpful clues in authorship attribution, while it is more unlikely that the frequencies of use of content-bearing words may be of help. This shows that choosing the right dimensions for the right task requires a deep understanding, on the part of the engineer, of the nature of the task.

### Reducing the Dimensionality of the Vectors

The techniques described in the previous section tend to generate very large vectors, frequently in the tens of thousands. While such a situation is not problematic in text search, whose standard algorithms are fairly robust with respect to the dimensionality of the vectors, it is in TC, since the efficiency of many learning devices (e.g., neural networks) tends to degrade rapidly with the size of the vectors. In TC applications, it is thus customary to run a dimensionality reduction pass before starting to build the internal representations of the documents. This means identifying a new vector space in which to represent the documents in such a way that the new vectors have a much smaller number of dimensions than the original ones. Several techniques for dimensionality reduction have been devised within TC (or, more often, borrowed from the fields of machine learning and pattern recognition).

An important class of such techniques is feature extraction methods (e.g., term clustering methods, latent semantic indexing). Feature extraction methods define a new vector space in which each dimension is a combination of some or all of the original dimensions; their effect is usually a reduction of both the dimensionality of the vectors and the overall stochastic dependence among dimensions.

An even more important class of dimensionality reduction techniques is that of feature selection methods, which do not attempt to generate new terms, but try to select the best ones from the original set. The measure of quality for a term is its expected impact on the accuracy of the resulting classifier. To measure this, feature selection functions are employed for scoring each term according to this expected impact so that the highest scoring terms can be retained for the new vector space. These functions mostly come from statistics (e.g., chi-

square and information theory). Mutual information, also known as information gain tend to encode each in their own way the intuition that the best terms for classification purposes are the ones that are distributed most differently across the different categories.

## CHALLENGES

Text categorization, especially in its machine learning incarnation, is now a fairly mature technology that has delivered working solutions in a number of applicative contexts. Still, a number of challenges remain for TC research.

The first and foremost challenge is delivering high accuracy in *all* applicative contexts. While highly effective classifiers have been produced for applicative domains such as the thematic classification of professionally authored texts (such as newswires), in other domains reported accuracies are far from satisfying. Such applicative contexts include the classification of Web pages, where the use of text is more varied and obeys rules different from those of linear verbal communication, spam filtering, a task that has an adversarial nature; in that spammers adapt their spamming strategies to circumvent the latest spam filtering technologies; and authorship attribution, in which current technology is not yet able to tackle the inherent stylistic variability among texts written by the same author.

A second important challenge is to bypass the *document labeling bottleneck* (i.e., the fact that labelling (namely, manually classifying), documents for use in the training phase is costly).

To this end, semisupervised methods have been proposed that allow building classifiers from a small sample of labelled documents and a usually larger sample of unlabelled documents (Nigam, McCallum, Thrun, & Mitchell, 2000). However, the problem of learning text classifiers mainly from unlabelled data is still, unfortunately, open.

## REFERENCES

- Diederich, J., Kindermann, J., Leopold, E., & Paaß, G. (2003). Authorship attribution with support vector machines. *Applied Intelligence*, 19(1/2), 109-123.
- Drucker, H., Vapnik, V., & Wu, D. (1999). Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5), 1048-1054.
- Dumais, S. T., & Chen, H. (2000). Hierarchical classification of Web content. In N. J. Belkin, P. Ingwersen, & M.-K. Leong (Eds.), *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval* (pp. 256-263). Athens, Greece: ACM Press.
- Fall, C. J., Törösvári, A., Benzineb, K., & Karetka, G. (2003). Automated categorization in the International Patent Classification. *SIGIR Forum*, 37(1), 10-25.
- Giorgetti, D., & Sebastiani, F. (2003). Multiclass text categorization for automated survey coding. In *Proceedings of SAC-03, 18th ACM Symposium on Applied Computing* (pp. 798-802). Melbourne, Australia: ACM Press.
- Hayes, P. J., & Weinstein, S. P. (1990). Construe/Tis: A system for content-based indexing of a database. *Innovative Applications of Artificial Intelligence* (pp.49-66). Menlo Park, CA: AAAI Press.
- Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In C. Nédellec & C. Rouveirol (Eds.), *Proceedings of ECML-98, 10th European Conference on Machine Learning* (pp. 137-142). Chemnitz, DE: Springer Verlag.
- Koppel, M., Argamon, S., & Shimon, A. R. (2002). Automatically categorizing written texts by author gender. *Literary and Linguistic Computing*, 17(4), 401-412.
- Maron, M. (1961). Automatic indexing: An experimental inquiry. *Journal of the Association for Computing Machinery*, 8(3), 404-417.
- Nigam, K., McCallum, A. K., Thrun, S., & Mitchell, T. M. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3), 103-134.
- Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? Sentiment classification using machine learning techniques. *Proceedings of EMNLP-02, 7th Conference on Empirical Methods in Natural Language Processing* (pp. 79-86). Philadelphia: Association for Computational Linguistics.
- Schapire, R. E. & Singer, Y. (2000). BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3), 135-168.
- Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1), 1-47.
- Stamatatos, E., Fakotakis, N., & Kokkinakis, G. (2000). Automatic text categorization in terms of genre and author. *Computational Linguistics*, 26(4), 471-495.



### KEY TERMS

**Boosting:** One of the most effective types of learners for text categorization. A classifier built by boosting methods is actually a committee (or ensemble) of classifiers, and the classification decision is made by combining the decisions of all the members of the committee. The members are generated sequentially by the learner, which attempts to specialize each member by correctly classifying the training documents the previously generated members have misclassified most often.

**Classifier:** An algorithm that, given as input two or more classes (or labels), automatically decides to which class or classes a given document belongs, based on an analysis of the contents of the document. A single-label classifier is one that picks one class for each document. When the classes among which a single-label classifier must choose are just two, it is called a binary classifier. A multilabel classifier is one that may pick zero, one, or many classes for each document.

**Dimensionality Reduction:** A phase of classifier construction that reduces the number of dimensions of the vector space in which documents are represented for the purpose of classification. Dimensionality reduction beneficially affects the efficiency of both the learning process and the classification process. In fact, shorter vectors need to be handled by the learner and by the classifier, and often on the effectiveness of the classifier too, since shorter vectors tend to limit the tendency of the learner to “overfit” the training data.

**Learner (or Supervised Learning) Algorithm:** A general inductive process that automatically generates a classifier from a training set of preclassified documents.

**Supervised (Machine) Learning:** A form of machine learning (i.e., improving the machine’s performance at

performing a task by exposing the machine to experiential data). A learning method is supervised when it relies on the exposure to preclassified data, that is, to data items that have previously been labelled by classes (or categories) from a predefined finite set.

**Support Vector Machines:** One of the most effective types of learners for text categorization. They attempt to build a classifier that maximizes the margin (i.e., the minimum distance between the hyperplane that represents the classifier and the vectors that represent the documents). Different functions for measuring this distance (kernels) can be plugged in and out; when nonlinear kernels are used, this corresponds to mapping the original vector space into a higher-dimensional vector space in which the separation between the examples belonging to different categories may be accounted for more easily.

**Terms:** The dimensions of the vector space in which documents are represented according to the vector space model. In thematic applications of text categorization, terms usually coincide with the content-bearing words (or with their “stems”) that occur in the training set, while in nonthematic applications they may be taken to coincide with the topic-neutral words or with other custom-defined global characteristics of the document.

**Vector Space Model:** A popular method for representing documents and determining their semantic relatedness, originally devised in the mid 1960s for text search applications and subsequently applied in the representation of documents for text categorization applications. Documents are represented as vectors in a vector space generated by the terms that occur in a document corpus (the document collection in text search, the training set in text categorization), and semantic relatedness is usually measured by the cosine of the angle that separates the two vectors.



# Text Databases

**Gonzalo Navarro**

*University of Chile, Chile*

## INTRODUCTION

A *text* is any sequence of symbols (or *characters*) drawn from an alphabet. A large portion of the information available worldwide in electronic form is actually in text form (other popular forms are structured and multimedia information). Some examples are natural language text (e.g., books, journals, newspapers, jurisprudence databases, corporate information, the Web), biological sequences (e.g., ADN and protein sequences), continuous signals (e.g., audio and video sequence descriptions, time functions), and so on. Recently, due to the increasing complexity of applications, text and structured data are being merged into so-called semistructured data, which is expressed and manipulated in formats such as XML (out of the scope of this review).

As more text data is available, the challenge to search them for queries of interest becomes more demanding. A *text database* is a system that maintains a (usually large) text collection and provides fast and accurate access to it. These two goals are relatively orthogonal, and both are critical if one is to profit from the text collection.

Traditional database technologies are not well suited to handle text databases. Relational databases structure the data in fixed-length records whose values are atomic and are searched for by equality or ranges. There is no general way to partition a text into atomic records, and treating the whole text as an atomic value is of little interest for the search operations required by applications. The same problem occurs with multimedia data types, hence the need for specific technology to manage texts.

The simplest query to a text database is a string pattern: The user enters a string and the system points out all the positions of the collection where the string appears. This simple approach is insufficient for some applications in which the user wishes to find a more general *pattern*, not just a plain string. Patterns may contain wild cards (i.e., characters that may appear zero or more times), optional characters that may appear in the text or not, classes of characters (i.e., string positions that match a set of characters rather than a single one), gaps that match any text substring within a range of lengths, and so forth. In many applications, patterns must be *regular expressions*, composed by simple strings and union, concatenation, or repetition of other

regular expressions. These extensions are typical of computational biology applications but also appear in natural language searches.

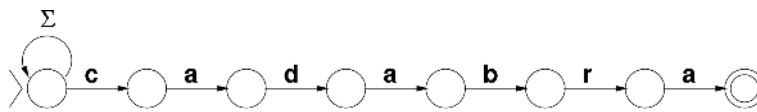
In addition, a text database may provide *approximate searching*, that is, means for recovering from different kinds of *errors* that the text collection (or the user query) may contain. There may be, for example, typing, spelling, or optical character recognition errors in natural language texts; experimental measurement errors or evolutionary differences in biological sequences; and noise or irrelevant differences in continuous signals. Depending on the application, the text database should provide a reasonable error model that permits users to specify some tolerance in their searches. A simple error model is the *edit distance*, which gives an upper limit to the total number of character insertions, deletions, and substitutions needed to match the user query with a text substring. More complex models give lower costs to more probable changes in the sequences.

The previously mentioned searches are collectively called *syntactic search*, because they are expressed in terms of sequences of characters present in the text. In natural language texts, *semantic search* is of great value. In this case, the user expresses an information need and the system retrieves portions of the text collection (i.e., documents) that are relevant to that need, even if the query words do not directly appear in the answer. A model that determines the relevance of a document with respect to a query is necessary. This model *ranks* the documents and offers the highest ranked documents to the user. Unlike in the syntactic search, there are no right or wrong answers, just better and worse ones. However, because the user's time spent browsing the results is the most valuable resource, good ranking is essential in the success of the search.

## SYNTACTIC SEARCH

In syntactic searching, the desired outcome of a search pattern is clear, so the main concern of a text database is efficiency. Two approaches are possible with pattern matching: sequential and indexed searching (these can be combined, as we will see). Sequential searching assumes that it is not possible to preprocess the text, so only the pattern is preprocessed and then the whole text

Figure 1. An NFA to recognize strings terminated in pattern “cadabra”



database is sequentially scanned. Indexed searching, on the other hand, preprocesses the text so as to build a data structure of it (an “index”), which can be used later to speed up searches. Building an index is usually a time and memory demanding task, so indexed searching is possible only when several conditions are met: (a) The text must be large enough to justify it against the simpler sequential search, (b) the text must change rarely enough so as to amortize the indexing work over many queries, and (c) there must be enough space to hold the index, which can be quite large in some applications.

## Sequential Searching

### Simple String Matching

In the simplest search problem, we are given a string pattern  $P$  of  $m$  characters, a text  $T$  of  $n$  characters (which may be the concatenation of all texts in the collection), and are required to point out all the occurrences of  $P$  in  $T$  (Navarro & Raffinot, 2002). There exist many string matching algorithms. The most famous are Knuth-Morris-Pratt (KMP), for being the first in achieving the optimal  $O(n)$  worst-case time; and Boyer-Moore (BM), for being the first in achieving so-called sublinear search time, meaning that it does not inspect every text character (actually, its search time improves as  $m$  grows). Another famous algorithm is Backward DAWG Matching (BDM), for its average-case optimality,  $O(n \log_{\sigma}(m)/m)$ , where  $\sigma$  is the alphabet size if we assume a uniformly distributed text. Many other algorithms are relevant to theorists for their algorithmic features: optimal in space usage, bounded time to access the next text character, simultaneous worst- and average-case optimality, and so on.

In practice, the most efficient string-matching algorithms derive from their theoretically more appealing variants. Shift-Or can be seen as a derivation of KMP that is twice as fast and  $O(n)$  in most practical cases. Horspool is a simplification of BM that is usually the fastest in searching natural language text. Backward Nondeterministic DAWG Matching (BNDM) is a derivation of BDM that is faster to search biological sequences of moderate length. Backward Oracle Matching (BOM) also derives from BDM and is the fastest for longer biological sequences.

The use of finite automata is at the kernel of pattern matching. Many of the previously mentioned algorithms rely on automata. Figure 1 shows a nondeterministic finite automaton (NFA) that reaches its final state every time the string it has consumed finishes with a given pattern  $P$ . If fed with the text, this automaton will signal all the endpoints of occurrences of  $P$  in  $T$ . Algorithm KMP consists essentially in making that NFA deterministic (a DFA), and using it to process each text character in  $O(n)$  time. Shift-Or, on the other hand, uses the automaton directly in nondeterministic form. The NFA states are mapped to bits in a computer word, and a constant number of logical and arithmetic operations over the computer word simulate the update of all the NFA states simultaneously, as a function of a new text character. The result is twice as fast as KMP in practice, and  $O(n)$  provided  $m \leq w$ , with  $w$  being the number of bits in the computer word (32 or 64 nowadays). For  $m > w$ , Shift-Or is still  $O(n)$  on average. This technique to map NFA states to bits is called *bit-parallelism*, and it has been extensively used in pattern matching to simulate NFAs directly without converting them to DFAs.

In Figure 2 we see an NFA that recognizes every reversed *prefix* of a pattern  $P$ , that is, a prefix of  $P$  read backwards (a prefix is a substring that starts at the beginning of the string). Note that the NFA will have active states as long as we have fed it with a reversed substring of  $P$ . This automaton is used by BDM and BNDM, the former by converting it to deterministic and the latter with bit-parallelism. The idea is to build such an NFA over the reverse pattern. Given a text *window* (segment of length  $m$ ) where the pattern could appear, we feed the automaton with the window characters read right to left. If the automaton runs out of active states at some point, it means that the window suffix we have read is not a substring of  $P$ , so  $P$  cannot appear in any window containing the suffix read. Hence, we can shift the window over  $T$  to the position following the window character that killed the automaton. Actually, we can remember the last time the automaton signaled a prefix of  $P$  and shift to align the window to that position. If, on the other hand, we read the whole window and the automaton still has active states, then  $P$  is in the window, so we report it and shift to align the window to the last prefix recognized (see Figure 3, left side). Algorithm BOM is also based on determinizing this NFA, albeit in

Figure 2. An NFA to recognize the suffixes of “cadabra,” reversed

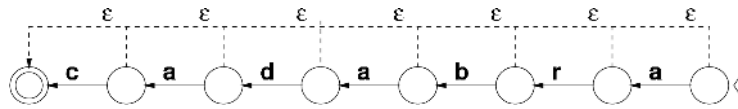
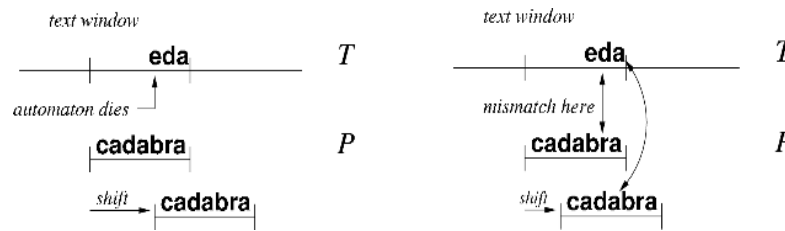


Figure 3. On the left, the BDM search process in a window. On the right, the same window processed by Horspool



an imperfect way that gives shorter shifts but runs faster because it is simpler.

Horspool algorithm is not based on automata. It simply compares the current text window against  $P$ , reports an occurrence in case of equality, and then shifts the window so that its last character  $c$  aligns with the last occurrence of  $c$  in  $P$ . This does not lose any possible occurrence and wins because of its simplicity in all but small alphabets (see Figure 3, right side).

time exponential in  $m$ , which is disturbing in practice even for moderate sized patterns. Another approach is to simulate the NFA directly using bit-parallelism. Very efficient bit-parallel simulations of the restricted extended patterns mentioned previously have been developed. When simulating general regular expressions, however, the same exponential dependency on  $m$  appears. It is possible, using bit-parallelism, to search in  $O(mn/\log(s))$  time given  $O(s)$  space for the preprocessing.

### Extended Patterns and Regular Expressions

Patterns with wild cards, optional characters, gaps, and, in general, regular expressions, can be searched for by extending algorithms designed for simple strings (Navarro & Raffinot, 2002). The prevailing approach is, again, based on automata. Figure 4 shows some examples. A regular expression can be converted into an NFA and then a DFA, which can be used for searching in  $O(n)$  worst-case time. The problem is that this may need space and

### Approximate Pattern Matching

It is usually convenient to allow some tolerance in pattern matching against the text, independently of the pattern searched for. The usual approach is to define a *distance* between strings that models the desired tolerance and let the user specify a tolerance threshold  $k$  together with  $P$ . Then, the system reports text substrings whose distance to  $P$  does not exceed  $k$  (Navarro, 2001; Navarro & Raffinot, 2002).

Figure 4. On top, an NFA to recognize an extended pattern with optional and repeatable characters. On the bottom, the NFA of a general regular expression.

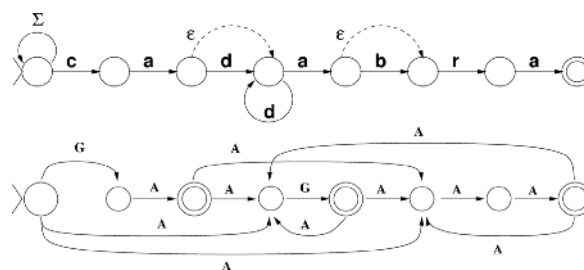
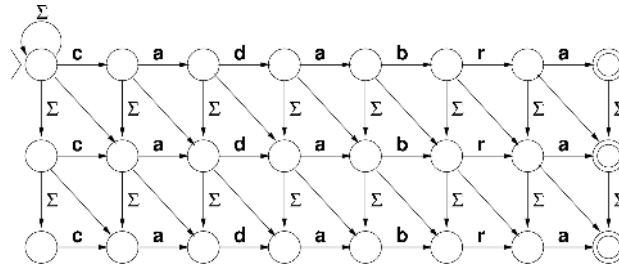


Figure 5. An NFA to find “cadabra” within edit distance at most 2. Unlabeled arrows are traversed by  $\Sigma$  and  $e$



The most general technique for approximate pattern matching relies on dynamic programming. A virtual matrix is filled such that at cell  $(i, j)$  contains the minimum distance between the length- $i$  prefix of  $P$  and a substring of  $T$  finishing at position  $j$ . Each cell can be filled in constant time, giving  $O(mn)$  worst-case search time.

For some simple (and popular) distance functions, such as the edit distance, much faster algorithms exist. As before, is it possible to express the search as the outcome of an automaton, as depicted in Figure 5. This NFA can be made deterministic so as to search in  $O(n)$  worst-case time. As for regular expressions, the problem is that the preprocessing time and space is exponential in  $m$  or  $k$ , which makes the DFA approach unpopular. Bit-parallelism, however, can be used to simulate the NFA directly so as to obtain very practical  $O(kmn/w)$  worst-case time algorithms. Moreover, the dynamic programming matrix can be computed in bit-parallel so as to obtain  $O(mn/w)$  time.

Another useful approach for the case of low  $k/m$  values (which are the most interesting in practice) is to *filter* the text to discard most of it with a necessary condition that can be checked quickly. For example, if  $P$  is cut into  $k + 1$  nonoverlapping pieces, then any occurrence within edit distance  $k$  must contain one of the pieces unaltered. A multipattern search for the pieces followed by classical verification of areas around the

occurrences is very simple and one of the fastest algorithms, with average complexity  $O(nk \log_o(m)/m)$ . For small alphabets, the best algorithms in practice are also optimal on average:  $O(n(k+\log_o(m))/m)$ .

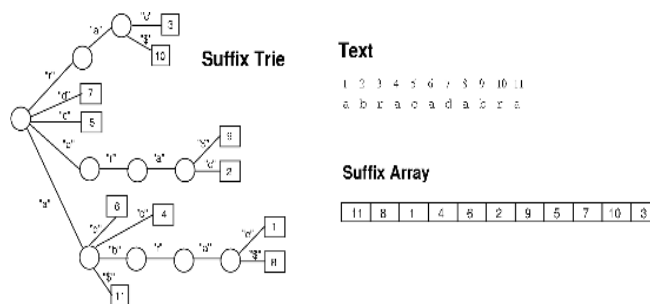
Several of these techniques can also be adapted to approximate searching of regular expressions and extended patterns, with good results.

### Indexed Searching

When the text can be preprocessed to build an index, the most popular general-purpose choices are suffix trees and arrays (Gusfield, 1997). The best choice for the specific case of natural language is the inverted index, reviewed later in this article.

A *digital tree* or *trie* is a tree where each branch is labeled by a character, and the children of a node are labeled by different characters. The set of strings represented by a trie are obtained by reading all its root-to-leaf paths and concatenating the characters labeling each path. A *suffix trie* is a trie representing all the suffixes of a given text. At the leaves, we store the starting position of the suffixes. Note that, because every substring is the prefix of some suffix, every text substring can be found by following a path from the suffix trie root. Hence, in order to search for all the occurrences of  $P$  in  $T$ , it is enough to traverse the suffix tree path of  $T$  corre-

Figure 6. The suffix trie and suffix array for the text “abracadabra”



sponding to the characters of  $P$ . All the occurrences of  $P$  are then found at the leaves of the subtree. This requires optimal time  $O(m+occ)$  to retrieve the  $occ$  occurrences of  $P$  (see Figure 6).

Although optimal at searching, the suffix trie can be of quadratic size in the worst case. The *suffix tree* is a variant that guarantees  $O(n)$  space and construction time, making the overall scheme optimal in theory. In practice, even the suffix tree may be too large (10 to 20 times the text size), so more compact structures may be preferred. The most popular is the *suffix array*, which contains just the leaves of the suffix tree, or which is the same, the sequence of pointers to all text suffixes in lexicographic order. The suffix array requires four times the text size, which is more manageable, though still large. Just as all the occurrences of  $P$  appear in a single subtree of the suffix tree, they appear in a single interval of the suffix array, which can be binary searched at  $O(m \log n)$  time. Very recently, compression techniques for suffix arrays have obtained great success, but the techniques are still immature.

Suffix trees and arrays can be searched for complex patterns, too (Navarro, Baeza-Yates, Sutinen, & Tarhio, 2001). The general idea is to define an automaton that recognizes the desired pattern, and backtrack in the suffix tree (or simulate a backtrack on the array), going through all the branches. The traversal stops either when the automaton runs out of active states (in which case the subtree is abandoned) or when it reaches a final state (in which case the subtree leaves are reported). On most regular expressions and approximate searching,  $O(n^\lambda)$  worst-case time, for some  $0 < \lambda < 1$ , is achieved.

Natural language texts, at least in most Western languages, have some remarkable features that make them easier to search (Baeza-Yates & Ribeiro-Neto, 1999; Witten, Moffat, & Bell, 1999). The text can be seen as a sequence of *words*. The number of different words grows slowly with the text size. The query is usually a word or a sequence of words (a phrase). In this case, *inverted indexes* are a low-cost solution that provide very good performance. An inverted index is just a list of all the different words in the text (the *vocabulary*), plus a list of the occurrences of each such word in the text collection. A search for a word reduces to a vocabulary search (with hashing, for example) plus outputting the list. A search for a complex pattern that matches a word is solved by sequential scanning over the vocabulary plus outputting the matching word lists. A phrase search is solved by intersecting the occurrences of the involved words. An inverted index typically needs 15% to 40% of the space of the text, but several compression techniques to the index and the text can be successfully applied (Witten, Moffat, & Bell, 1999).

## SEMANTIC SEARCH

In semantic searching, the text collection is seen as a set of *documents*, and the goal is to establish the *relevance* of each document with respect to a user query, so as to present the most relevant ones to the user. The query can be a set of words or phrases, or even another document. As accuracy in the answers is a central concern, the choice of a *model* to compute how relevant is a document with respect to a query is fundamental (Baeza-Yates & Ribeiro-Neto, 1999).

### Relevance Models

The most popular relevance model is called the *vector model*, in which each document is seen as a vector in a very high dimensional space. Each coordinate represents a word of the collection vocabulary. The value of the document vector along a coordinate is given by the relevance of the corresponding word to distinguish that document from others. The query is also seen as a vector, and the cosine between two vectors is their similarity measure.

A popular measure of the relevance of a word to distinguish a document from others is  $tf \times idf$ , where  $tf = f/mf$ ,  $f$  is the number of times the word appears in the document,  $mf$  is the number of times the most frequent word of the document appears in it,  $idf = \log(N/n)$ ,  $N$  is the total number of documents, and  $n$  is the number of documents where the word appears. For Web searching, this model is usually enriched with the information given by the links between Web pages so that pages pointed to from “relevant” pages are also considered relevant.

Another less popular model, which is usually mixed with the vector model, is the boolean model. In this model, documents are relevant or not relevant. The query specifies which words must appear in relevant documents, which words must not appear, and indicates intersections, unions and complementation of resulting document sets. There are several other less popular models of relevance.

### Evaluation

An issue related to relevance computation is how to evaluate the accurateness of the results. The most popular measures are called *precision* and *recall*. Precision indicates which fraction of the retrieved documents are actually relevant. Recall indicates which fraction of the relevant documents were actually retrieved. A human judgement is necessary to establish which are the docu-



ments really relevant to the query. Note that it is easy to have higher precision by retrieving fewer documents and higher recall by retrieving more documents, since the ranking just orders the document but does not indicate how many to retrieve. Hence the correct measure is a “precision-recall plot,” where the precision and recall obtained when retrieving more and more documents are drawn in a two-dimensional plot (one coordinate for precision and the other for recall). For example, one can plot the precision obtained for recall values of 0%, 10%, ..., 100%.

## Indexes

Variants of the inverted index are useful for semantic searches as well. These indexes indicate only the documents where each word appears, as well as its number of occurrences. This is useful to compute term  $f$  in the vector ranking. Moreover, because only the few highest ranked documents will be output, documents with highest  $f$  value will be the most interesting ones, so the lists of documents where each term appears are sorted by decreasing  $f$  value. Several heuristics are used to avoid processing long lists of probably irrelevant documents. Although these may alter the correct outcome of the query, the relevance model is already a guess of what the user wants, so it is permissible to distort it a bit in exchange for large savings in efficiency. Note that this tradeoff is not permitted in syntactic searching.

## FUTURE TRENDS

Textual databases face the following challenges in upcoming years:

- The increasing size of text collections, which in cases such as the Web, reach several terabytes. Maintaining efficiency and accuracy of results under this scenario is much more complex than for medium-sized collections.
- The increasing complexity of search operations, typical of applications in computational biology and other areas. This poses the need to find more complex patterns, which are usually much more difficult to search for than simple strings.
- The low quality of the data, with different reasons depending on the application: no quality control in the Web, experimental errors in DNA sequences, and so forth. This poses the need for approximate matching methods well-suited to each application.
- The size of the indexes, which, especially for general texts, is more relevant than the search speed. In a

large text, such an index requires frequent access to secondary memory, which is much slower than main memory.

- The need to deal with secondary memory, which is a rather undeveloped feature in some indexes, especially those for general texts. Some indexes still perform poorly in secondary memory.
- The need to cope with changes in the text collection, as text indexes usually require expensive rebuilding upon text changes.
- The need to provide, for text databases, the same level of data management achieved for atomic values in traditional databases, with respect to powerful query languages (e.g., permitting joins), concurrency, recovery, security, and so forth.

## REFERENCES

- Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval*. Reading, MA: Addison-Wesley.
- Gusfield, D. (1997). *Algorithms on strings, trees and sequences: Computer science and computational biology*. Cambridge, UK: Cambridge University Press.
- Navarro, G. (2001). A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1), 31-88.
- Navarro, G., Baeza-Yates, R., Sutinen, E., & Tarhio, J. Indexing methods for approximate string matching. *IEEE Data Engineering Bulletin*, 24(4), 19-27.
- Navarro, G., & Raffinot, M. (2002). *Flexible pattern matching in strings: Practical on-line search algorithms for texts and biological sequences*. Cambridge, MA: Cambridge University Press.
- Witten, I., Moffat, A., & Bell, T. (1999). *Managing gigabytes* (2nd Ed.). San Francisco: Morgan Kaufmann.

## KEY TERMS

**Approximate Searching:** Searching that permits some differences between the pattern specification and its text occurrences.

**Bit-Parallelism:** A technique to store several values in a single computer word so as to process them all at once.

**Edit Distance:** A measure of string similarity that counts the number of character insertions, deletions and substitutions needed to convert one string into the other.

**Indexed Searching:** Searching with the help of an index, a data structure previously built on the text.

**Precision and Recall:** Measures of retrieval quality, relating the documents retrieved with those actually relevant to a query.

**Sequential Searching:** Searching without preprocessing the text.

**Suffix Trees and Arrays:** Text indexes that permit fast access to any text substring.

**Semantic Search:** A search in natural language text, based on the meaning rather than the syntax of the text.

**Syntactic Search:** A text search based on string patterns that appear in the text, without reference to meaning.

**Text Database:** A database system managing a collection of texts.

**Vector Model:** A model to measure the relevance between a document and a query, based on sharing distinctive words.

# Transaction Concurrency Methods

Lars Frank

Copenhagen Business School, Denmark

## INTRODUCTION

In this article, I will evaluate the most important methods for increasing concurrency between transactions. Because different transactions have different needs, it is important that the evaluation gives an overview of the advantages and disadvantages of the different optimization methods.

It is possible to increase concurrency between database transactions and minimize the time in which conflicts may occur by splitting major transactions into minor transactions performing the same operations. For short, I will call this concurrency optimization technique *short-duration locking*, even though not all concurrency control methods use locking. Concurrency may also be increased by using by using *low isolation levels*, in which certain types of conflicts between transactions are accepted. In practice, either *two-phase locking* or *optimistic concurrency control* is used for concurrency control. When one of these concurrency control methods is chosen, the only way to increase concurrency further is to relax the isolation property. In practice, the isolation property is relaxed by using low isolation levels, short duration locks, or both. Therefore, these optimization methods are analyzed, too.

By using the results of my analyses, it should be easier to increase concurrency between database transactions. This is especially important in long-lived transactions, hotspots, and multidatabases in which the traditional concurrency control methods often have to give up in practice. However, when either short duration locking or low isolation levels optimize concurrency, the traditional atomicity, consistency, isolation, and durability (ACID) properties are normally lost. The problems caused by a lack of ACID properties may be managed by using approximated ACID properties (i.e., from an application point of view the system should function as if all the traditional ACID properties had been implemented).

When the isolation property is relaxed, the following isolation anomalies may occur (Berenson et al., 1995; Breibart, 1992):

- *The lost update anomaly* is by definition a situation in which a first transaction reads a record for update without using locks. After this, the record is

updated by another transaction. Later, the update is overwritten by the first transaction. In practice, the read and the update of a record is often executed in different subtransactions to prevent locking the record across a dialogue with the user. In this situation, it is possible for a second transaction to update the record between the read and the update of the first transaction. If countermeasures are not used, the update of the second transaction may be lost.

- *The dirty read anomaly* is by definition a situation in which a first transaction updates a record without committing the update. After this, a second transaction reads the record. Later, the first update is aborted (i.e., the second transaction may have read a nonexisting version of the record). In extended transaction models, this may happen when the first transaction updates a record by using a compensatable subtransaction and later aborts the update by using a compensating subtransaction. If a second transaction reads the record before it is compensated, the data read will be “dirty.”
- *The nonrepeatable read anomaly* or *fuzzy read* is by definition a situation in which a first transaction reads a record without using *long duration locks*. This record is later updated and committed by a second transaction before the first transaction is committed or aborted. In other words, one cannot rely on what one has read. In extended transaction models, this may happen when the first transaction reads a record updated by a second transaction, which commits the record locally before the first transaction commits it globally.
- *The phantom anomaly* is normally not important and, therefore, I do not deal with it in this article.

## BACKGROUND

Different concurrency methods and isolation levels have been analyzed and evaluated (e.g., Gray & Reuter, 1993), and short duration locks have been used in practice for many years. However, to my knowledge, short duration locks have not been evaluated and compared to other methods in terms of increasing concurrency between transactions.

The transaction model described in the next section is the *countermeasure transaction model* described by Frank and Zahle (1998), Frank (1999, 2004), and Frank and Kofod (2002). This model owes many of its properties to, for example, Garcia-Molina and Salem (1987); Mehrotra, Rastogi, Korth, and Silberschatz (1992); Weikum and Schek (1992); and Zhang, Nodine, Bhargava, and Bukhres (1994). The countermeasure transaction model (Frank & Zahle, 1998) is important because it is the first systematic attempt to find countermeasures against isolation anomalies caused by using short duration locking. In this article, I have generalized this idea and included new countermeasures that especially apply to isolation level anomalies.

Kempster, Stirling, and Thanisch (1999) have described isolation anomalies in detail. The objective of this article is not to evaluate concurrency control methods as such, as this has been done by Thomasian (1998), for example.

In the following, I will first describe how the atomicity property can be implemented by using extended transaction models. This is important if short duration locks are used. Next, I will describe and evaluate some of the methods most used to increase concurrency between transactions. The results of the evaluation will either be explained or adopted from references.

## ATOMICITY IMPLEMENTATION FOR TRANSACTIONS THAT USE SHORT DURATION LOCKS

An updating transaction has the *atomicity property* and is called *atomic* if either all or none of its updates are executed. If short duration locks are used, the atomicity property cannot be implemented automatically by the DBMS, and therefore extended transaction models are recommended. In extended transaction models, a global transaction consists of a *root transaction* (client transaction) and several single-site *subtransactions* (server transactions). The subtransactions can be nested transactions (i.e., a subtransaction may be a *parent transaction* for other subtransactions). In the following, I will give a broad outline of how the atomicity property is implemented in extended transaction models.

In extended transaction models, a global transaction is partitioned into the following types of subtransactions that may be executed in different locations:

- *The Pivot* subtransaction manages the atomicity of the global transaction (i.e., the global transaction is committed when the pivot subtransaction is committed locally). If the pivot subtransaction aborts,

all the updates of the other subtransactions must be compensated or not be executed.

- *Compensatable* subtransactions, that all may be compensated. Compensatable subtransactions must always be executed *before* the pivot subtransaction, to make it possible to compensate them if the pivot subtransaction cannot be committed. Compensation is achieved by executing a *compensating* subtransaction.
- *Retriable* subtransactions are designed so that the execution is guaranteed to commit locally (sooner or later) if the pivot subtransaction is committed. An update propagation tool is used to automatically resubmit the request for execution until the subtransaction has been committed locally (i.e., the update propagation tool is used to force the retrievable subtransaction to be executed).

Executing compensatable, pivot, and retrievable subtransactions, in that order, implements the global atomicity and atomicity property of a transaction (Breitbart et al., 1992).

## METHODS TO INCREASE CONCURRENCY

In this section, I will analyze and evaluate pessimistic concurrency control, optimistic concurrency control, low isolation levels, and short duration locks as different methods to increase concurrency. Table 1 gives an overview of the properties of the methods used to increase concurrency.

### Pessimistic Concurrency Control

Locking protocols implements serializability by using locks (i.e., each transaction locks the data it uses to prevent other transactions from accessing *its* data). Locking protocols can be described as *pessimistic*, because they assume that data used by one transaction may be needed by other transactions and, therefore, it is better to lock the data.

Pessimistic concurrency control can be optimized by using shared and exclusive locks for reading and writing/updating, respectively. All locks should have *long duration* (i.e., they must be held until the transaction has been committed). However, the designer of the transaction may divide it into subtransactions that manage the locks independently of the parent transaction. In this way, the programmer can force *short duration* locks to be used. If short duration locks are used, approximated ACID properties should be used, as described previ-

## Transaction Concurrency Methods

ously. *Deadlock* occurs when two or more transactions are in a simultaneous waiting position, whereby each is waiting for the other to release a lock before it can proceed. By using short duration locks, it is possible to prevent and reduce the number of deadlocks, because the probability of deadlock diminishes with the number of records locked in a subtransaction.

### Optimistic Concurrency Control

*Optimistic concurrency control* assumes that data used by one transaction normally will not be needed by other concurrent transactions, and therefore all transactions have access to all data. However, at commit time a conflict occurrence, is checked.. If a conflict has occurred, the offending transaction is restarted. Otherwise, all updates of the transaction are written to the database and the transaction is committed. Optimistic concurrency control does not use locks because all transactions have access to all data until commit time. By dividing a transaction into subtransactions, it is possible to increase concurrency as the time in which conflicts may occur is reduced. However, if subtransactions are used, approximated ACID properties should be used in the same way as described previously.

Franaszek and Robinson (1985) show that optimistic concurrency control normally gives more concurrency between transactions than locking protocols. However, in a *hotspot*, in which a record is updated by many concurrent transactions, pessimistic concurrency control is generally better than optimistic concurrency control (O'Neil, 1986).

### Low Isolation Levels

The *isolation level* of a transaction for a table is defined as the degree of interference tolerated on the part of concurrent transactions. Therefore, it is possible to define many different isolation levels. Table 2 defines the relationship between the isolation levels of the SQL standard and the isolation anomalies that are accepted for a given isolation level.

The lower the isolation level of a transaction, the more concurrency is allowed and the more different isolation anomalies may occur for the transaction. Therefore, to achieve high performance and availability, it is important to select as low an isolation level as the applications can tolerate. The isolation levels defined by the SQL standard can primarily increase the concurrency in *read-write conflicts*, whereby transactions read re-

Table 1. Evaluation overview of methods to increase concurrency

Evaluation criteria	Pessimistic concurrency control	Optimistic concurrency control	Low isolation levels	Short duration locks
<b>Concurrency</b>	Allows concurrency except when conflicting locks occur	Allows concurrency except for conflicting transactions	Concurrency is improved at the costs of "non-lost-update" anomalies	Concurrency is improved at the costs of all types of isolation anomalies
<b>Deadlock</b>	When deadlock occurs one of the transactions must be aborted	Eliminates deadlock, but one of the transactions must be aborted in case of a conflict	Can eliminate deadlocks caused by read-write conflicts	Can reduce/eliminate dead-locks caused by both read-write and write-write conflicts
<b>Hotspots</b>	Pessimistic concurrency is generally better than optimistic concurrency control in hotspots	Pessimistic concurrency is generally better than optimistic concurrency control in hotspots	Can eliminate read-write conflicts in hotspots	Can reduce both read-write and write-write conflicts in hotspots
<b>Atomicity property</b>	No problems	No problems	No problems	Extended transaction models <i>should</i> be used
<b>Consistency property</b>	No problems	No problems	Countermeasures may be used to manage isolation anomalies	Asymptotic consistency <i>should</i> normally be used
<b>Isolation property</b>	No problems except for long-duration transactions	No problems except for long-duration transactions	Countermeasures may be used to manage isolation anomalies	Countermeasures <i>should always</i> be used to manage isolation anomalies
<b>Durability property</b>	No problems	No problems	No problems	No problems if the atomicity property is implemented
<b>Distribution options</b>	Distributed concurrency control will decrease performance and availability	Distributed concurrency control will decrease performance and availability	May be implemented in a distributed DBMS without further problems	Retriable subtransactions may be necessary to implement the atomicity property
<b>Development costs</b>	A DBMS facility	A DBMS facility	Extra costs for countermeasures against isolation anomalies	Extra development costs for approximated ACID properties



Table 2. Anomalies allowed by the different isolation levels of SQL

SQL isolation levels	Isolation anomalies		
	Dirty read	Nonrepeatable read	Phantoms
Serializable	No	No	No
Repeatable read	No	No	Yes
Read committed	No	Yes	Yes
Read uncommitted	Yes	Yes	Yes

spectively write and update a table. As described later, this is not as good as *short duration locking*, which can reduce or eliminate both read–write and write–write conflicts.

Countermeasures against the isolation anomalies enable applications to tolerate isolation anomalies to a certain degree. In the next section, I will describe countermeasures against the isolation anomalies that may occur when short duration locks are used. In principle, these countermeasures can also be used against anomalies caused by low isolation. However, countermeasures against lost updates are an exception as long as low isolation levels do not allow lost updates.

In the following, I will describe countermeasures that can be used to design transactions that can tolerate the isolation anomalies caused by selecting low isolation levels.

*The semantic analysis countermeasure* analyzes the semantic meaning of the data in context of its use to relax isolation. For example, from a semantic point of view, dirty data are probably correct, because the data normally would have been committed if an error had not occurred and caused the update to be aborted. Therefore, it is normally safe to read dirty data if the data cannot be misused (e.g., selling from empty stocks). On the other hand, it is normally risky to read nonrepeatable data because the data are out of date and therefore might be wrong. In other words, data that cannot be misused should have an isolation level in which dirty reads are accepted and nonrepeatable reads are forbidden. Unfortunately, such an isolation level does not yet exist in the SQL standard. However, a great deal of research into the foundations of isolation levels has been done (Kempster et al., 1999), and some DBMS products do not respect the isolation level limitations of the SQL standard.

Alternatively, if data with a low isolation level can be misused, the data must be protected by higher isolation, provided that other countermeasures cannot give sufficient protection.

*The restricted use countermeasure* limits the use of dirty and nonrepeatable read data. For example, this type of data must not be written to the database either directly or in a modified form, as the value of the data may be inconsistent. The data may be used for decision-

making if the data are updated using the pessimistic view countermeasure or if the value of the data is not time critical. However, a user should know that, in both cases, the data might be inconsistent with other data stored in the database.

## Short Duration Locks

In distributed systems, transactions often consist of a root transaction in a client location that manages several subtransactions in one or more servers. Each time the root transaction receives new information from the user, it starts a new subtransaction in a server until the transaction is finished. This type of transaction is *long lived*, as the user may be relatively slow in making his or her input, and other concurrent users do not want to wait for the first user to finish his or her job before they can access the same data. This is especially the case if the first user is interrupted or lacks information to finish his or her job. Therefore, short duration locks are often used in practice. *Short duration locks* are local locks that are released as soon as possible (i.e., data will, for example, not be locked across locations). In systems with relatively many updates, such as ERP-systems and e-commerce systems, low isolation levels cannot solve the availability problem, as all update locks must be exclusive. In such situations, I recommend using short duration locks and an extended transaction model that solves the problems with the missing ACID properties. Next, I will describe some of the most important countermeasures against isolation anomalies that occur when the isolation property is reduced by short duration locking.

*The reread countermeasure* is primarily used to prevent the lost update anomaly. Transactions that use this countermeasure read a record twice, using short duration locks for each read. If a second transaction has changed the record between the two readings, the transaction aborts itself after the second read. Often, it is not acceptable to lock records across a dialogue with a user, as no one knows how long it will take the user to make the input data. In such a situation, the reread countermeasure may be used to protect against the lost update anomaly in the following way:

## Transaction Concurrency Methods

First, the application reads all the records needed using short duration shared locks, which are released after the readings. Next, the user enters data for the updating process assuming that no other transaction will change the records. After the user has entered the data, the application rereads the records, this time using short duration exclusive locks. The update will only be executed and committed if the record has not been changed since it was read the first time. (An update time-stamp in all database records might detect whether data has been changed since it was read the first time). The reread countermeasure may prevent the lost update anomaly and the dirty read anomaly in compensatable and pivot subtransactions. The reread countermeasure is the most used measure in practice.

The *commutative updates countermeasure* can prevent lost updates merely by using commutative operators. Adding and subtracting an amount from an account are examples of commutative updates. If a subtransaction only has commutative updates, it may be designed as commutable with other subtransactions that also only have commutative updates. This is very important because compensatable, compensating, and retrievable subtransactions have to be commutative to enable the countermeasure to prevent the lost update anomaly. The commutative updates countermeasure has earlier been described in, e.g. Sagas (Garcia-Molina and Salem, 1987) and *Open Nested Transactions* (Weikum and Schek, 1992), where the countermeasure is used in both central and distributed databases.

The *pessimistic view countermeasure against short duration locks anomalies* reduces or eliminates the dirty read anomaly, the nonrepeatable read anomaly, or both by giving the users a pessimistic view of the situation. In other words, the user cannot misuse the information. The purpose is to eliminate the risk involved in using data in which long duration locks should have been used. A pessimistic view countermeasure may be implemented by using the following:

- The pivot or compensatable subtransactions for updates that limit the users' options, and
- The pivot or retrievable subtransactions for updates that increase the users' options.

## FUTURE TRENDS

In the current SQL standard, it is possible for a program only to select an isolation level for itself. However, a program may know if it is updating risky data, and, therefore, it should also be possible for a program to force other programs to accept a minimum of locking rules (i.e.,

isolation levels) while it is running. The number of isolation levels of the SQL standard is also very limited, and there is no reason for ordering them in a hierarchy. By dropping all these restrictions, it is possible to both increase concurrency and find many more countermeasures to manage the risky anomaly situations. Especially when short duration locking is used, it is important that an application can force other applications to accept a minimum of locking rules, as the application may know that the risk of seeing the update depends on whether some "stock" has been increased or diminished.

Recently, there have been major developments within replication methods. How to manage concurrency should be integrated in the decision of how to manage recovery and data replication.

## CONCLUSION

In this article, I have evaluated the most important methods for increasing concurrency between transactions. Because different transactions have different needs, it is important that the evaluation gives an overview of the advantages and disadvantages of the different optimization methods.

Short duration locking and low isolation levels are mandatory for *long-lived transactions*, as long duration locks per definition cannot be recommended for such transactions. As all transactions with a user dialogue are long lived, it is important to know how these techniques can be used to increase concurrency.

The countermeasure transaction model (Frank & Zahle, 1998) is important, as it is the first systematic attempt to find countermeasures against isolation anomalies caused by using short duration locking. In this article, I have generalized this idea and included new countermeasures that especially apply to isolation-level anomalies. I have also described how the different optimization methods can be integrated. This is important, as short duration locking and the use of low isolation levels are independent methods to increase concurrency, and therefore they can support each other. I hope that my analysis will make it easier to increase concurrency between database transactions in practice.

## REFERENCES

Berenson, H., Bernstein, P., Gray, J., Melton, J., O'Neil, E., & O'Neil, P. (1995). A critique of ANSI SQL isolation levels. *Proceedings of the ACM SIGMOD Conference* (pp. 1-10).

Breibart, Y., Garcia-Molina, H., & Silberschatz, A. (1992). Overview of multidatabase transaction management. *VLDB Journal*, 2, 181-239.

Franaszek, P., & Robinson, J. (1985). Limitations in concurrency in transaction processing. *ACMTODS*, 10(1).

Frank, L. (1999). Evaluation of the basic remote backup and replication methods for high availability databases. *Software - Practice & Experience*, 29(15), 1339-1353.

Frank, L. (2004). Transaction design for databases with high performance and availability. *Journal of Informaiton and Organizational Sciences*, 28(1-2), 41-47.

Frank, L., & Kofod, U. (2002). Atomicity implementation in e-commerce systems. *Proceedings of the Second International Conference on Electronic Commerce*, Taipei, Taiwan, Republic of China.

Frank, L., & Zahle, T. (1998). Semantic ACID properties in multidatabases using remote procedure calls and update propagations. *Software - Practice & Experience*, 28, 77-98.

Garcia-Molina, H., & Salem, K. (1987). Sagas. *Proceedings of the ACM SIGMOD Conference* (pp. 249-259).

Gray, J., & Reuter, A. (1993). *Transaction processing*. Morgan Kaufman.

Hadzilacos, V., & Toueg, S. (1993). Fault-tolerant broadcasts and related problems. In S. Mullender (Ed.), *Distributed systems* (pp. 97-145). Addison-Wesley.

Kempster, T., Stirling, C., & Thanisch, P. (1999). Diluting ACID properties. *SIGMOD Record*, 28(4).

Mehrotra, S., Rastogi, R., Korth, H., & Silberschatz, A. (1992). A transaction model for multi-database systems. *Proceedings of the International Conference on Distributed Computing Systems* (pp. 56-63).

O'Neil, P. (1986). The escrow transaction mode. *ACM TODS*, 11(4).

Thomasian, A. (1998). Concurrency control: Methods, performance, and analyses. *ACM Computing Surveys*, 30(1).

Weikum, G., & Schek, H. (1992). Concepts and applications of multilevel transactions and open nested transactions. In A. Elmagarmid (Ed.), *Database transaction models for advanced applications* (pp. 515-553). Morgan Kaufmann.

Zhang, A., Nodine, M., Bhargava, B., & Bukhres, O. (1994). Ensuring relaxed atomicity for flexible transactions in multidatabase systems. *Proceedings of the ACM SIGMOD Conference* (pp. 67-78).

## KEY TERMS

**ACID Properties:** The properties imply that a transaction is *atomic*, transforms the database from one *consistent* state to another consistent state, is executed as if it were *isolated* from other concurrent transactions, and is *durable* after it has been committed.

**Atomic Transaction:** A transaction the updates of which are either all executed or removed (the "A" in the ACID properties). The atomicity property makes it easier to do database recovery.

**Concurrency Control:** Control method, which secures that a transaction is executed as if it were executed in isolation (the "I" in the ACID properties) from other concurrent transactions.

**Hotspots:** Records that are updated so frequently that they constitute bottlenecks if traditional concurrency control is used.

**Isolation Anomalies:** Inconsistencies that occur when transactions are executed without the isolation property.

**Isolation Levels:** Different degrees of isolation that use up to three different types of isolation anomalies, which are accepted when concurrent transactions are executed.

**Long-Lived Transactions:** Transactions that run for so long that they will block access to the records they have accessed if traditional concurrency control is used.

**Multidatabases:** Integrated databases managed by local database management systems without distributed ACID properties.

**Transaction Abort:** Removal of the updates of a transaction.

**Two-Phase Locking:** The most used concurrency control method for implementing the isolation property.

# Transactional Support for Mobile Databases

Hong Va Leong

The Hong Kong Polytechnic University, Hong Kong

## INTRODUCTION

With the widespread deployment of wireless communication infrastructure in the past decade, accessing information online while a client is on the move becomes a concrete possibility. Such a computing environment is often referred to as a *mobile environment* (Imielinski & Badrinath, 1994). A typical group of applications that deserve strong support under the mobile environment would be database access. Database systems that support operations initiated from mobile clients are referred to as *mobile databases* (Leong & Si, 1997). We have witnessed a tremendous growth in mobile database research in the past ten years. Yet only the most primitive results have been incorporated in real applications. This is due to the additional dimensions of complexity that the mobile environment has introduced, beyond standard client/server computing environment.

Besides traditional issues to accessing a database system under a client/server setting, a mobile environment also suffers from problems such as low communication bandwidth and unreliable communication with occasional disconnection. To combat for the disconnection problem of mobile clients, database server will allow clients to cache data items that they may need in future, either performing caching on demand or prefetching cache in an anticipatory manner (Jing, Helal & Elmagarmid, 1999). In the context of the mobile file system Coda, this latter approach is called hoarding (Mummert, Ebling & Satyanarayanan, 1995).

There are claims that a mobile environment does possess a high broadcast bandwidth. It is this peculiar feature of *asymmetric communication* that leads to a good number of research works. For instance, it is quite intuitive to utilize the high bandwidth downlink channel to schedule useful data items to be broadcast to a large collection of mobile clients, thereby improving the scalability of database access, since the same bandwidth can serve a large client population (Si & Leong, 1999).

With downscaling of mobile clients from laptop computers to PDAs and even to smart phones, the relatively weak processing power of mainstream mobile clients cannot be solved in a satisfactory manner without considering a certain tradeoff. To reduce the processing stress on the small mobile clients, it is necessary to move the computation from the clients back to

the server. Under such environment, the clients often only implement the interface, passing the data back to the server for operation. However, one may not be willing to give up the autonomy of mobile clients by relinquishing the processing control to the server. The most acceptable variant that still favors client control is through the use of *mobile agents* (Yau, Leong & Si, 2003), which can act as surrogates on behalf of the mobile clients in performing possibly complex operations on the database and to prepare for handling processing results upon client disconnection. The captured results would only be conveyed back to the mobile clients upon reconnection.

## BACKGROUND

Access needs from mobile clients to database server in a mobile environment should be backed by efficient mechanisms, which are specially designed for those mobile databases. To further complicate matters, clients in a mobile environment can physically move around. As a result, the database applications should be enriched by taking into consideration the client location as a special form of data in the database. As such, that gives rise to the need of providing client location management to be tracked through the use of a special kind of databases, namely, *moving object databases* (Wolfson et al., 1999). Location tracking and management performance can be improved by providing approximate moving object locations, which often suffice for most applications (Lee, Leong & Si, 2003). With object location in light, a new class of database queries whose result sets would depend on the client location need to be handled. These are often referred to as location-dependent queries (Madria et al., 2000).

Most common applications accessing mobile databases often access a collection of related data items for information purpose, that is, for inquiry. There are often only few updates to the databases. In such cases, it may be easy to provide a consistent view of data items to mobile clients, since the updates can be batched and installed to the databases in the background. This is also the philosophy behind many mobile file systems (such as Coda) that only provide the session semantics for remote accesses. However, there are certain applications that require a strong level of consistency to access-



ing a related set of data items, satisfying the referential integrity (accessing no phantom tuple) or other integrity constraints such as the mutual consistency of the item set (e.g., the total number of seats sold plus remaining seats should be constant in a show). This can be guaranteed via the execution of *transactions*, satisfying the four conditions of Atomicity, Consistency, Isolation, and Durability (Bernstein, Hadzilacos & Goodman, 1987). The widely accepted correctness criterion of executing concurrent transactions is the *serializability* of transactions. Transactions initiated by mobile clients that are executed on mobile databases are called *mobile transactions* (Dirckze & Gruenwald, 2000; Mok, Leong & Si, 1999).

Extending standard ACID mobile transactions further, there are occasions that one needs to access data not only from one database, but also from multiple databases residing on different sites, perhaps spanning different administrative domains. It becomes more difficult to ensure consistency, especially when the databases are managed by different organizations. The extension of serializability in such a context is called the *global serializability* (Breitbart, Garcia-Molina & Silberschatz, 1992), and the transaction spanning across different organizations is known as a *global transaction*. Tesch and Wäsch (1997) presented an implementation of global transactions on the ODMG-compliant multidatabase systems. Preserialization techniques were proposed to improve performance of global transactions in the mobile environment (Dirckze & Gruenwald, 2000).

Although global transactions ensure strong consistency, the cost of global transaction execution is very high, so they are not widely adopted in practice. Instead, a global transaction is often split into multiple smaller *subtransactions* for execution on individual databases. Each subtransaction can commit by itself under the Saga concept (Garcia-Molina & Salem, 1987). Such a collection of a logically related global transaction can be derived from the concept of a *nested transaction* (Moss, 1987). Alternatively, *compensating transactions* (Chrysanthis & Ramamritham, 1994) can be executed to undo the effect of failed subtransactions. It has been argued that execution of compensating transactions is often more appropriate in a mobile environment to reduce blocking (Tesch & Wäsch, 1997), since mobile transactions are relatively long-lived.

## SUPPORTING MOBILE TRANSACTIONS

Transactional semantics to access a database satisfying the serializability of transactions can be enforced by means of *concurrency control protocols*, such as two-

phase locking and timestamp ordering (Bernstein et al., 1987). Most practical database systems employ the strict two phase locking protocol as their concurrency control mechanism. In the context of a mobile database, it is often more common to implement a variant of the optimistic concurrency control protocol (Bernstein et al., 1987) and two-phase locking with lock caching (Franklin, Carey & Livny, 1997) in view of the high communication overhead. The benefits of optimistic concurrency control protocol lie in the reduction of communication overhead in acquiring locks, thereby deferring the detection of conflicts to the end of a transaction. This is appropriate under the assumption that data conflict is rare.

To improve the performance of variants of optimistic concurrency control protocol as applied in the mobile environment, especially when the degree of conflict is not very low, the server may issue a *certification report* periodically to mobile clients (Barbara, 1997). The certification report contains data items belonging to the read set and write set of active transactions that have successfully been certified for commitment. In other words, the report conveys information about conflicts in data items from transactions that have been validated. Mobile clients that execute transactions conflicting with those in the certification report can choose to abort those transactions that are destined to fail. To further facilitate early detection of conflicts, relevant information can be conveyed to clients when the latter issue requests for data items. This enables early partial validation to be performed at the server (Lee & Lam, 1999) since the read set of the requesting mobile transaction is known. Another way to reduce the number of mobile transactions being aborted is to reduce the time interval between the availability of updated values of data items to the time the corresponding update transactions commit. This is resolved by introducing the *pre-write operation* that allows early availability of updated values (Madria & Bhargava, 2001). The corresponding *pre-read operation* allows return of the value updated by a pre-write operation.

A very special property in a mobile environment is the availability of the high bandwidth broadcast media. As a result, database items can be scheduled to be broadcast over the downlink channel, thereby allowing efficient processing of transactions, especially those *read-only transactions* (Pitoura & Chrysanthis, 1999). However, one must ensure that the set of data items constituting a broadcast cycle are mutually consistent by taking a *consistent snapshot* of the database. Furthermore, an invalidation report can be broadcast at the beginning of a broadcast cycle to allow a mobile client to abort transactions that have read an old data value, which has subsequently been overwritten (i.e., reading



outdated value). Read-only transaction performance can be further improved by means of multi-version broadcast, since reading older and yet consistent versions can avoid invalidation caused by update transactions. It is simple to construct a consistent snapshot of broadcast items by executing a large read-only transaction. However, this large long-lived read-only transaction will generate excessive blocking to concurrent update transactions. To improve efficiency, the snapshot is often taken by means of some special internal database mechanism. This action must be handled with great care in the presence of update transactions, since inconsistency can easily arise (Ammann, Jajodia & Mavuluri, 1995).

To process update transactions in a broadcast environment where data items are broadcast, updates must be conveyed back to the server via the uplink channel. As such, it is beneficial to classify the data items into hot items and cold items. Hot data items commonly needed by many clients can be scheduled for broadcast in order to capitalize on the high broadcast bandwidth, whereas cold data items can be requested on demand via the uplink channel. The hybrid protocol by Mok et al. (1999) takes advantage of the broadcast channel for delivering a consistent snapshot of hot items while making use of the uplink channel to request for items not available over the broadcast. The protocol ensures serializability by performing validation for update transactions against values stored at the database server. As with previous work, updates are installed at the server in an atomic step with short-duration conservative two-phase locking. Consistency across database items is ensured through the use of timestamps, and update transactions are serialized in timestamp order.

Although the use of broadcast with multi-versioning can improve commit rate of read-only transactions, further performance improvement can be gained by relaxing the strong serializability requirement a little bit. It is intuitive that mobile clients are not interested in observing a *global serialization order* on all transactions. Instead, they are only concerned about their views on transactions that they can see to be guaranteed consistent. This is captured in the notion of *update consistency* (Shanmugasundaram et al., 1999) in which all update transactions are serializable, but each mobile client only needs to ensure that the projection of transactions that it can observe is serializable. Update consistency can be enforced by the cycle-based algorithm through the use of F-matrix. Values of committed updates to data items are broadcast in each cycle, together with control information that allows a mobile client to determine the consistency of the set of data items that it reads. While update consistency can guarantee the logical consistency of transactions with respect to the server and each mobile client, *isolation-only transaction* (Lu & Satyanarayanan,

1995) was proposed to reduce the impact of client disconnection by focusing on the isolation property of transactions. In isolation-only transaction, the new second class transactions can be executed and conditionally committed under a disconnected environment. Upon reintegration, it is possible that those second class transactions can be converted to first class transactions if there are no conflicts (unlikely) or the conflicts can be repaired semantically. Second class transactions could miss all non-local transactions, but first class transactions are serializable among themselves.

Going one step further from update consistency and isolation-only transactions, which still guarantee consistency under favorable conditions, it is often acceptable for a mobile client not to see the updates made by concurrently executing mobile clients, if transactions can be executed at a higher efficiency. In this regard, one can relax the isolation property of a transaction that it is allowed to miss at most  $N$  concurrently executing transactions under the correctness criterion of  $N$ -ignorance (Krishnakumar & Bernstein, 1994). Intuitively,  $N$  concurrent transactions can be executed without any concurrency control. If more transactions are executed, only limited constraints need to be imposed. Observing that it is the deviation of data values from the correct values that is more important than the actual number of transactions ignored during an execution, it would be more acceptable to control the degree of deviation of data values returned by a read operation. As long as the deviation on a data item  $x$  is within a prescribed threshold  $\epsilon$ , conflicting weaker transactions (called  $\epsilon$ -transactions) on  $x$  can still commit under the correctness criterion of  $\epsilon$ -serializability (Wu, Yu & Pu, 1997) through the use of divergence control methods. Wong, Agrawal, and Mak (1997) generalized the notion of controlled deviation into *bounded inconsistency*. Instead of considering only the simple read and write operations, general operations such as “increment” or “insert”, executed by concurrent conflicting transactions, return data values in the form of a *resolution set*. As long as the resolution set satisfies the correctness criteria indicating the allowable deviation, conflicting transactions can be committed with bounded amount of inconsistency. As a result, operations can be executed out of order in some controlled manner to improve concurrency and hence performance.

## FUTURE TRENDS

With the rapid advancement in communication devices in recent years, it is anticipated that smart phones and PDAs will overtake laptop computers to become the predominated mobile devices. They now possess in-

creasing processing power and storage capacity. Smart phones are programmable so that they approach simple mobile clients in terms of functionality. Furthermore, with the increasing popularity of Web services such as .NET and SOAP, new mobile applications would likely be built upon such platforms. All these will change the landscape of accesses to mobile databases, including transactional accesses. Although it is likely still based on the client-server access paradigm so that most existing concurrency control protocols would work, there are recurring limitations on mobile clients in terms of connectivity and battery life. As such, it is reasonable that some of the complex coordination work would be migrated to the proxy server or to Web services components, thereby alleviating the processing need on the mobile clients. Transaction processing mechanisms, including concurrency control protocols, would more likely be performed at the proxy server and be decoupled from the application logic. The application logic can further be delivered conveniently through the adoption of mobile agents (Yau et al., 2003) that can migrate with respect to client movement and may even hop from devices to devices. On the other hand, more research and development efforts would be dedicated to intelligent agents that can make sensible decisions on behalf of the client, especially in the event of disconnection. In particular, mobile agents are highly useful for event-driven transactions like stock selling transactions or auctioning, since they can make proper decisions without suffering from the delay through the wireless network for user input and the disconnection problem. Compensating transactions and global transactions for intelligent agents should also be considered. Since some compensating transactions expect user intervention, proper actions performed by the agent in the lack of physical user presence would be required.

## CONCLUSION

In this paper, we have presented the characteristics of mobile databases and their inherent limitation and difference from traditional databases. In particular, the difference in communication bandwidth, asymmetric communication infrastructure, disconnection as well as physical client movement issues need to be addressed. We focused more on one important class of applications on the databases, namely, atomic database accesses in terms of transactions, which guarantee consistent accesses across different data items. To generalize the transactions, we presented the notion of global transactions, which guarantee consistency not only across data items, but also across multiple database systems, perhaps even under different administrative domains.

Transactions can be supported through appropriate concurrency control protocols to ensure their correctness. In the context of mobile transactions, traditional concurrency control protocols should be adapted for the low bandwidth environment with disconnection. Furthermore, it is important to capitalize on the scalable broadcast channel to deliver data items from database to improve performance. Weaker consistency requirements for transactions were also proposed to allow performance to be boosted at the expense of allowing a controlled degree of inconsistency. In the future, we expect the rise of smaller mobile devices that would depend more heavily on Web services and the use of agents with transactional support provided at proxy.

## REFERENCES

- Ammann, P., Jajodia, S., & Mavuluri, P. (1995). On-the-fly reading of entire databases. *IEEE Transactions on Knowledge and Data Engineering*, 7(5), 834-838.
- Barbara, D. (1997). Certification reports: Supporting transactions in wireless systems. *Proceedings of the 17th International Conference on Distributed Computing Systems* (pp. 466-473).
- Bernstein, P.A., Hadzilacos, V., & Goodman, N. (1987). *Concurrency control and recovery in database systems*. Reading, MA: Addison-Wesley.
- Breitbart, Y., Garcia-Molina, H., & Silberschatz, A. (1992). Overview of multidatabase transaction management. *VLDB Journal*, 1(2), 181-239.
- Chrysanthis, P.K., & Ramamritham, K. (1994). Synthesis of extended transaction models using ACTA. *ACM Transactions on Database Systems*, 19(3), 450-491.
- Dirckze, R.A., & Gruenwald, L. (2000). A pre-serialization transaction management technique for mobile multidatabases. *Mobile Networks and Applications Journal*, 5(4), 311-321.
- Franklin, M.J., Carey, M.J., & Livny, M. (1997). Transactional client-server cache consistency: Alternatives and performance. *ACM Transactions on Database Systems*, 22(3), 315-363.
- Garcia-Molina, H., & Salem, K. (1987). Sagas. *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 249-259).
- Imielinski, T., & Badrinath, B.R. (1994). Mobile wireless computing: Challenges in data management. *Communications of the ACM*, 37(10), 18-28.

- Jing, J., Helal, A.S., & Elmagarmid, A. (1999). Client-server computing in mobile environments. *ACM Computing Surveys*, 31(2), 117-157.
- Krishnakumar, N., & Bernstein, A.J. (1994). Bounded ignorance: A technique for increasing concurrency in a replicated system. *ACM Transactions on Database Systems*, 19(4), 586-625.
- Lee, C.K., Leong, H.V., & Si, A. (2003). Approximating object location for moving object database. *Proceedings of the International Conference on Distributed Computing Systems Workshop on Mobile Distributed Computing* (pp. 402-407).
- Lee, V.C.S., & Lam, K.W. (1999). Optimistic concurrency control in broadcast environments: Looking forward at the server and backward at the clients. *Proceedings of First International Conference on Mobile Data Access*, (pp. 97-106).
- Leong, H.V., & Si, A. (1997). A semantic caching mechanism for mobile databases. *International Journal of Computers and Their Applications*, 4(2), 21-34.
- Lu, Q., & Satyanarayanan, M. (1995). Improving data consistency in mobile computing using isolation-only transactions. *Proceedings of the 5th Workshop on Hot Topics in Operating Systems*.
- Madria, S.K., Bhargava, B.K., Pitoura, E., & Kumar, V. (2000). Data organization issues for location-dependent queries in mobile computing. *Proceedings of International Conference on Database Systems for Advanced Applications* (pp. 142-156).
- Madria, S.K., & Bhargava, B.K. (2001). A transaction model to improve data availability in mobile computing. *Journal of Distributed and Parallel Databases*, 10(2), 127-160.
- Mok, E., Leong, H.V., & Si, A. (1999). Transaction processing in an asymmetric mobile environment. *Proceedings of First International Conference on Mobile Data Access* (pp. 71-81).
- Moss, J.E.B. (1987). Log-based recovery for nested transactions. *Proceedings of the International Conference on Very Large Data Bases* (pp. 427-432).
- Mummert, L.B., Ebling, M., & Satyanarayanan, M. (1995). Exploiting weak connectivity for mobile file access. *Proceedings of the 15th ACM Symposium on Operating System Principles* (pp. 143-155).
- Pitoura, E., & Chrysanthis, P.K. (1999). Scalable processing of read-only transactions in broadcast push. *Proceedings of the 19th International Conference on Distributed Computing Systems* (pp. 432-439).
- Shanmugasundaram, J., Nithrakashyap, A., Sivasankaran, R., & Ramamritham, K. (1999). Efficient concurrency control for broadcast environments. *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 85-96).
- Si, A., & Leong, H.V. (1999). Query optimization for broadcast database. *The Data and Knowledge Engineering Journal*, 29(3), 351-380.
- Tesch, T., & Wäsch, J. (1997). Global nested transaction management for ODMG-compliant multi-database systems. *Proceedings of International Conference on Information and Knowledge Management* (pp. 67-74).
- Wolfson, O., Sistla, A.P., Xu, B., Zhou, J., & Chamberlain, S. (1999). DOMINO: Databases fOr MovINg Objects tracking. *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 547-549).
- Wong, M.H., Agrawal, D., & Mak, H.K. (1997). Bounded inconsistency for type-specific concurrency control. *Journal of Distributed and Parallel Databases*, 5(1), 31-75.
- Wu, K.L., Yu, P.S., & Pu, C. (1997). Divergence control algorithms for epsilon serializability. *IEEE Transactions on Knowledge and Data Engineering*, 9(2), 262-274.
- Yau, S.M.T., Leong, H.V., & Si, A. (2003). Distributed agent environment: Application and performance. *Information Sciences Journal*, 154(1-2), 5-21.

## KEY TERMS

**Broadcast Database:** A broadcast database is a mobile database whose contents are being broadcast, fully or partially, to a population of mobile clients.

**Compensating Transaction:** A compensating transaction is a transaction that is executed to undo the effect of another committed transaction. Unlike ordinary transaction rollback or abort, both original and compensating transactions are visible in the committed projection of the execution history.

**Concurrency Control Protocol:** A concurrency control protocol is executed to ensure that the proper correctness criterion, usually serializability, is upheld for a set of concurrently executing transactions by controlling whether a certain operation can be performed, delayed, or

rejected and whether the transaction can be committed or has to be aborted.

**Database Snapshot:** A consistent collection of values of data items in a database that correspond to what a read-only transaction would collect. The snapshot can be used as a checkpoint for recovering a database upon a crash failure and for consistent database broadcast.

**Mobile Agent:** A mobile agent is a piece of code capable of migrating to different sites for execution, making use of local resources, and acting on behalf of its launching site. The agent will carry along with its execution state and data upon migration.

**Mobile Database:** A mobile database is a database accessible to mobile clients. There are appropriate mechanisms to take into account of the limitation of the wireless bandwidth, the use of downlink broadcast channel, and the effect of client mobility.

**Moving Object Database:** A moving object database is a database that maintains efficiently the location infor-

mation about moving objects with proper indexing on the object location to support advanced queries such as location-dependent queries and continuous queries.

**Serializability/Global Serializability:** Serializability is the generally accepted correctness criterion for concurrent execution of transactions. The concurrent execution should produce the same effect and lead to the same database state as one possible sequential execution of the same set of transactions. Global serializability is the correctness criterion for concurrent execution of global transactions over many database systems. It is a stronger correctness criterion than serializability.

**Transaction/Global Transaction:** A transaction is a sequence of operations on a database that should appear as if they were executed non-interfered, even in the presence of other concurrent transactions. A transaction should satisfy the ACID properties, namely, **A**tomicity, **C**onsistency, **I**solation, and **D**urability. A global transaction is a distributed transaction that is executed on two or more database systems.



# Transformation-Based Database Engineering

Jean-Luc Hainaut

University of Namur, Belgium

## INTRODUCTION

Modelling software design as the systematic transformation of formal specifications into efficient programs and building CASE tools that support it has long been considered one of the ultimate goals of software engineering. For instance, Balzer (1981) and Fikas (1985) consider that *the process of developing a program [can be] formalized as a set of correctness-preserving transformations [...] aimed to compilable and efficient program production*. In this context, according to Partsch and Steinbrüggen (1983), *a transformation is a relation between two program schemes  $P$  and  $P'$  (a program scheme is the [parameterized] representation of a class of related programs; a program of this class is obtained by instantiating the scheme parameters). It is said to be correct if a certain semantic relation holds between  $P$  and  $P'$* .

These definitions still hold for database schemas, which are a special kind of abstract program schemes. The concept of transformation is particularly attractive in this realm, though it has not often been made explicit (for instance, as a user tool) in current CASE tools.

A (schema) transformation is most generally considered to be an operator by which a data structure  $S1$  (possibly empty) is replaced by another structure  $S2$  (possibly empty) which may have some sort of equivalence with  $S1$ . Some transformations change the information contents of the source schema, particularly in schema building (adding an entity type or an attribute) and in schema evolution (removing a constraint or extending a relationship type). Others preserve it and will be called *semantics-preserving* or reversible.

Transformations that are proved to preserve the correctness of the original specifications have been proposed in practically all the activities related to schema engineering: schema normalization (Rauh & Stickel, 1995), DBMS schema translation (Rosenthal & Reiner, 1994), schema integration (McBrien & Poulouvasilis, 2003), schema equivalence (Jajodia, Ng, & Springsteel, 1983; Kobayashi, 1986), data conversion (Navathe, 1980), reverse engineering (Casanova & Amaral De Sa, 1984; Hainaut, Chandelon, Tonneau, & Joris, 1993), schema optimization (Halpin & Proper, 1995), database interoperability (McBrien & Poulouvasilis; Thiran & Hainaut, 2001), and others. The reader will find in

Hainaut (1995) an illustration of numerous application domains of schema transformations.

Though it has been explored for more than 25 years, this concept has only been gaining wider acceptance for two years, as witnessed by the recent references Omelayenko and Klein (2003) and van Bommel (2004).

The goal of this paper is to develop and illustrate a general framework for database transformations in which most processes mentioned above can be formalized and analyzed in a uniform way. Section 2 describes the basics of schema transformations. Section 3 explains how practical transformations can be used for database engineering. The database design and reverse engineering processes are revisited in Section 4, where we give them a transformational interpretation. Section 5 concludes the article.

## BACKGROUND

This section describes a general transformational theory that will be used as a basis for modelling database engineering processes. First, we define a wide-spectrum model from which operational models (i.e., those which are of interest for practitioners) can be derived. Then, we describe the concept of transformation and its semantics-preserving property.

### A Data Structure Specification Model

Database engineering is concerned with building, converting, and transforming database schemas at different levels of abstraction and according to various paradigms. Some processes, such as normalization, integration, and optimization operate within a single model and require intra-model transformations. Other processes, such as logical design, use two distinct models, namely, source and target. Finally, some processes—among others, reverse engineering and federated database development—can operate on an arbitrary number of models (or on a hybrid model made up of the union of these models) as we will see later on. The generic entity-relationship model (GER) is a wide-spectrum formalism intended to encompass most popular operational models, whatever their abstraction level and their underlying paradigms (Hainaut, 1996).



The GER includes, among others, the concepts of schema, entity type, entity collection, domain, attribute, relationship type, key, as well as various constraints. In this model, a schema is a description of data structures (Figure 1). It is made up of specification constructs, which can be, for convenience, classified into the usual three abstraction levels, namely, conceptual, logical, and physical:

- A *conceptual schema* comprises entity types, super/subtype (is-a) hierarchies, relationship types, roles, attributes (multi/single-valued; atomic/compound), identifiers (or unique keys), and various constraints.
- A *logical schema* comprises such constructs as record types, fields, arrays, foreign keys, redundant fields, etc.
- A *physical schema* comprises files, record types, fields, access keys (a generic term for index, calc key, etc), physical data types, bag and list multivalued attributes, and other implementation details.

Since it includes the main concepts of most operational models, the GER can be used to precisely specify each of them thanks to a *specialization* mechanism. According to the latter, each construct of model M is a specialization of a construct of the GER model. For instance, tables, columns, primary keys, and foreign

keys are specializations of, respectively, entity types, attributes, primary identifiers, and referential attributes. In addition, schemas in M must satisfy specific assembly rules, such as the following: *each entity type has at least one attribute*. As an important consequence, all intra- and inter-model transformations are specializations of GER-to-GER transformations. For instance, the standard ERA-to-SQL transformation is modelled by a chain of transformations from the ERA specialization of the GER to its SQL specialization.

The GER model has been given a formal semantics in terms of an extended NF2 model (Hainaut, 1996). This semantics allows us to analyze the properties of transformations and particularly to precisely describe how and under which conditions they propagate and preserve the information contents of schemas.

## Transformation: Definition

The definitions that will be stated are model-independent. In particular, they are valid for the GER model, so that the examples will be given in the latter. We denote by  $M$  the model in which the source and target schemas are expressed, by  $S$  the schema on which the transformation is to be applied, and by  $S'$  the schema resulting from this application.

A **transformation**  $\Sigma$  consists of two mappings  $T$  and  $t$  (Figure 2):

Figure 1. A typical hybrid schema made up of conceptual constructs (e.g., entity types PERSON, CUSTOMER, EMPLOYEE, and ACCOUNT; relationship type of; identifiers Customer ID of CUSTOMER); logical constructs (e.g., record type ORDER, with various kinds of fields, including an array, foreign keys ORIGIN and DETAIL.REFERENCE); and physical objects (e.g., table PRODUCT with primary key PRO\_CODE and indexes PRO\_CODE and CATEGORY, table space PRODUCT.DAT). The identifier of ACCOUNT, stating that the accounts of a customer have distinct Account numbers, makes ACCOUNT a dependent, or weak, entity type. The cardinality constraint of a role follows the participation interpretation and not the UML look-across semantics.

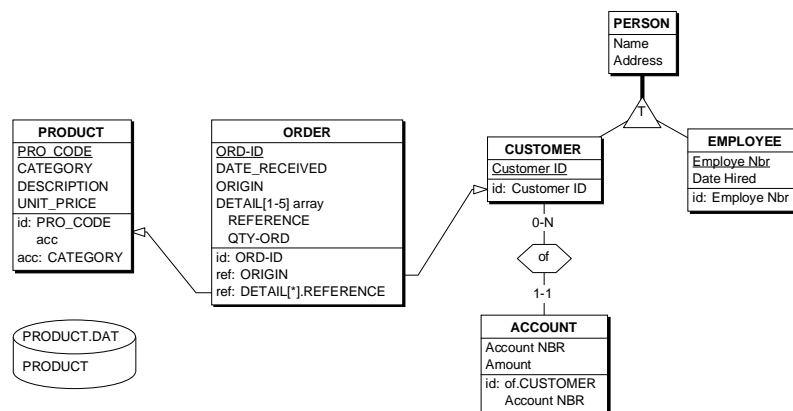
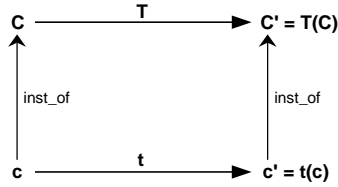


Figure 2. The two mappings of schema transformation  $\Sigma = \langle T, t \rangle$ . The *inst\_of* arrow from  $x$  to  $X$  indicates that  $x$  is an instance of  $X$ .



- **T** is the *structural mapping* that replaces source construct  $C$  in schema  $S$  with construct  $C'$ .  $C'$  is the target of  $C$  through  $T$  and is noted  $C' = T(C)$ . In fact,  $C$  and  $C'$  are classes of constructs that can be defined by structural predicates.  $T$  is therefore defined by the *minimal precondition P* that any construct  $C$  must satisfy in order to be transformed by  $T$ , and the maximal *postcondition Q* that  $T(C)$  satisfies.  $T$  specifies the rewriting rule of  $\Sigma$ .
- **t** is the *instance mapping* that states how to produce the  $T(C)$  instance that corresponds to any instance of  $C$ . If  $c$  is an instance of  $C$ , then  $c' = t(c)$  is the corresponding instance of  $T(C)$ . **t** can be specified through any algebraic, logical, or procedural expression.

According to the context,  $\Sigma$  will be noted either  $\langle T, t \rangle$  or  $\langle P, Q, t \rangle$ .

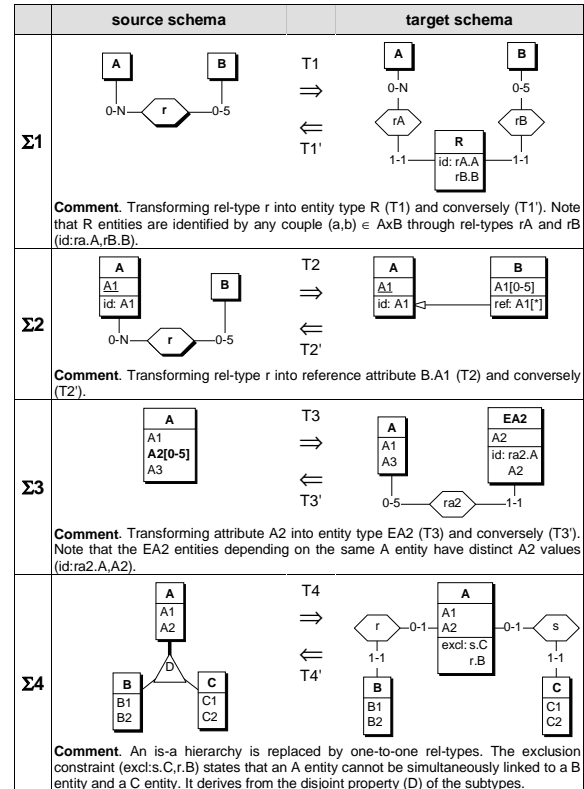
Each transformation  $\Sigma$  is associated with an inverse transformation  $\Sigma'$ , which can undo the result of the former under certain conditions.

### Reversibility of a Transformation

The extent to which a transformation preserves the information contents of a schema is an essential issue. Some transformations appear to augment the semantics of the source schema (e.g., adding an attribute), some remove semantics (e.g., removing an entity type), while others leave the semantics unchanged (e.g., replacing a relationship type with an equivalent entity type). The latter are called *reversible*, or *semantics-preserving*. If a transformation is reversible, then the source and the target schemas have the same descriptive power.

- A transformation  $\Sigma_1 = \langle T_1, t_1 \rangle = \langle P_1, Q_1, t_1 \rangle$  is *reversible*, iff there exists a transformation  $\Sigma_2 = \langle T_2, t_2 \rangle = \langle P_2, Q_2, t_2 \rangle$  such that for any construct  $C$  and any instance  $c$  of  $C$ :  $P_1(C) \Rightarrow ([T_2(T_1(C))]=C]$  and  $[t_2(t_1(c))=c]$ .  $\Sigma_2$  is the inverse of  $\Sigma_1$ , but the

Figure 3. The six mutation transformations  $\Sigma_1$  to  $\Sigma_3$ .  $\Sigma_4$  transforms an is-a hierarchy into one-to-one relationship types and conversely. The term *rel-type* stands for relationship type.



converse is not true. For instance, an arbitrary instance  $c'$  of  $T(C)$  may not satisfy the property  $c' = t_1(t_2(c'))$ .

- If  $\Sigma_2$  is reversible as well, then  $\Sigma_1$  and  $\Sigma_2$  are called *symmetrically reversible*. In this case,  $\Sigma_2 = \langle Q_1, P_1, t_2 \rangle$ .  $\Sigma_1$  and  $\Sigma_2$  are called *SR transformations* for short.

Thanks to the formal semantics of the GER, a proof system has been developed to evaluate the reversibility of a transformation (Hainaut, 1996).

## TRANSFORMATIONS FOR DATABASE ENGINEERING

In this section, some important basic transformations are described, as well as higher-level operators.

## Basic Transformations

A mutation is an SR transformation that changes the nature of an object. Considering the three main natures of object, namely, *entity type*, *relationship type*, and *attribute*, six mutation transformations can be defined (Figure 3). Mutations can solve many database engineering problems, but other operators are needed to model special situations.

Figure 3/bottom shows how a supertype/subtype hierarchy can be transformed by representing each source entity type by an independent entity type, then linking each subtype to its supertype through a one-to-one relationship type. The latter can, if needed, be further transformed into foreign keys by application of  $\Sigma 2$ -direct (T2). Field studies suggest that about 30 basic operators suffice for most engineering activities.

## Higher-Level Transformations

The transformations described so far are intrinsically atomic: One elementary operator is applied to one object instance, and none can be defined by a combination of others. The next sections describe two ways through which more powerful transformations can be developed.

## Predicate-Driven Transformations

A predicate-driven transformation  $\Sigma p$  applies an operator  $\Sigma$  to all the schema objects that meet a definite predicate  $p$ . It is specified by  $\Sigma(p)$ , where  $p$  is a *structural* predicate that states the properties through which a class of structures can be identified.

We give in Figure 4 some useful transformations that are expressed in the specific language of the DB-MAIN tool (Hainaut, Englebert, Henrard, Hick, & Roland, 1996), which follows the  $\Sigma(p)$  notation.

Most predicates are parametric; for instance, the predicate  $\text{ROLE\_per\_RT}(\langle n1 \rangle \langle n2 \rangle)$ , where  $\langle n1 \rangle$  and  $\langle n2 \rangle$  are integers, states that the number of roles of the relationship type falls in the range  $[\langle n1 \rangle .. \langle n2 \rangle]$ . The symbol “N” stands for infinity.

## Model-Driven Transformations

A model-driven transformation is a goal-oriented chain of predicate-driven operators. It is designed to transform any schema expressed in model  $M$  into an equivalent schema in model  $M'$ .

Identifying the components of a model also leads to identifying the constructs that do not belong to it. An arbitrary schema  $S$  expressed in  $M$  may include con-

Figure 4. Three examples of predicate-driven transformations.

predicate-driven transformation	interpretation
$\text{RT\_into\_ET}(\text{ROLE\_per\_RT}(3\ N))$	Transform each relationship type $R$ into an entity type ( $\text{RT\_into\_ET}$ ), if the number of roles of $R$ ( $\text{ROLE\_per\_RT}$ ) is in the range $[3\ N]$ ; in short, <i>convert all N-ary relationship types into entity types</i> .
$\text{RT\_into\_REF}(\text{ROLE\_per\_RT}(2\ 2)$ and $\text{ONE\_ROLE\_per\_RT}(1\ 2))$	Transform each relationship type $R$ into reference attributes ( $\text{RT\_into\_ET}$ ), if the number of roles of $R$ is 2 and if $R$ has from 1 to 2 one role(s), i.e., $R$ has at least one role with max cardinality 1; in short, <i>convert all one-to-many relationship types into foreign keys</i> .
$\text{INSTANTIATE}(\text{MAX\_CARD\_of\_ATT}(2\ 4))$	Transform each attribute $A$ into a sequence of single-value instances, if the max cardinality of $A$ is between 2 and 4; in short, <i>convert multivalued attributes with no more than 4 values into serial attributes</i> .

structs which violate  $M'$ . Each class of constructs that can appear in a schema can be specified by a structural predicate. Let  $PM$  denote the set of predicates that defines model  $M$  and  $PM'$  that of model  $M'$ . In the same way, each potentially invalid construct can be specified by a structural predicate. Let  $PM/M'$  denote the set of the predicates that identify the constructs of  $M$  that are not valid in  $M'$ . In the DB-MAIN language used in Figure 4,  $\text{ROLE\_per\_RT}(3\ N)$  is a predicate that identifies  $N$ -ary relationship types that are invalid in DBTG CODASYL databases, while  $\text{MAX\_CARD\_of\_ATT}(2\ N)$  defines multivalued attributes that are invalid in the SQL2 database model. Finally, we observe that  $PM$  can be perceived as a single predicate formed by *anding* its components.

Let us now consider predicate  $p \in PM/M'$ , and let us choose a transformation  $\Sigma = \langle P, Q \rangle$  such that:

$$(p \Rightarrow P) \wedge (PM' \Rightarrow Q)$$

Clearly, the predicate-driven transformation  $\Sigma p$  solves the problem of invalid constructs defined by  $p$ . Proceeding in the same way for each component of  $PM/M'$  provides us with a chain of operators that can transform any schema in model  $M$  into a schema in model  $M'$ . We express such a chain through a *transformation plan*, which is the practical form of any model-driven transformation.

If a plan comprises SR properties only, then the model-driven transformation that it implements is symmetrically reversible. When applied to any source schema, it produces a target schema semantically equivalent to the former. Figure 5 sketches a simple transformation plan intended to produce SQL2 logical schemas from ERA conceptual schemas. Actual plans are more complex but follow this approach.

Figure 5. A simple transformation plan to derive a relational schema from any ERA conceptual schema. To make them more readable, the transformations have been expressed in natural language instead of in the DB-MAIN language.

step	predicate-based transformation	comment
1	Transform IS-A relations into one-to-one rel-types.	Operator $\Sigma 4$ -direct.
2	Transform complex rel-types into entity types.	Operator $\Sigma 1$ -direct; <i>complex</i> means N-ary or binary many-to-many or with attributes.
3	Disaggregate level-1 compound attributes.	Each compound attribute directly depending on an entity type is replaced by its components.
4	Transform level-1 multivalued attributes into entity types.	Operator $\Sigma 3$ -direct; each multivalued attribute directly depending on an entity type is replaced by an entity type.
5	Repeat steps 3 to 4 until the schema does not include complex attributes anymore.	To cope with multilevel attribute structures.
6	Transform relationship types into reference groups.	At this point, only one-to-many and one-to-one rel-types subsist; they are transformed into foreign keys ( $\Sigma 2$ -direct).
7	If the schema still includes rel-types, add a technical identifier to the relevant entity types and apply step 6.	Step 6 fails in case of missing identifier; a technical attribute is associated with the entity type that will be referenced by the future foreign key.

Figure 6. Transforming a conceptual schema into a SQL2-compliant logical schema through the plan of Figure 5.

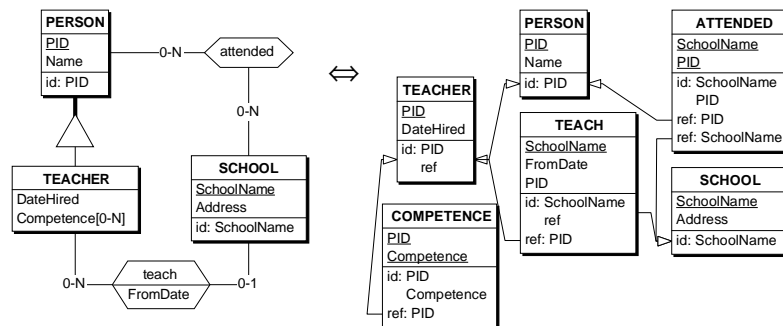


Figure 6 shows the application of this plan to a small ERA schema. Though model-driven transformations provide an elegant and powerful means of specification of many aspects of most database engineering processes, some other aspects still require human expertise that cannot be translated into formal rules.

## NEW PERSPECTIVES FOR DATABASE ENGINEERING PROCESS MODELLING

The transformational approach provides us with an elegant paradigm through which standard and nonstandard engineering processes can be revisited and built.

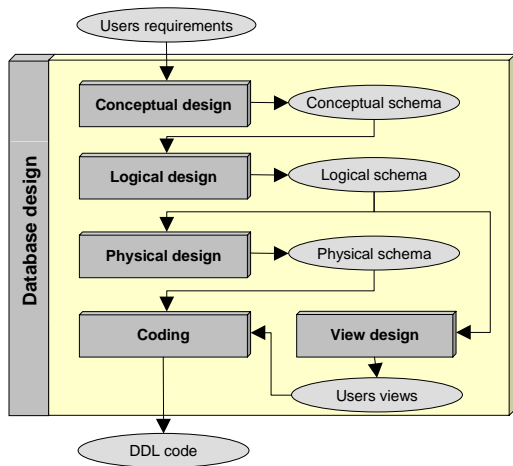
## Transformation-Based Database Design

Figure 7 describes the standard approach to database design as the transformation of users' requirements into DDL code. This transformation itself comprises five processes that, in turn, are transformations. Ignoring the view design process for simplicity, database design can be modelled by (the structural part of) transformation

DB-design:

DDL-code = DB-design(Users-requirements)

Figure 7. The standard strategy for database design



We can refine this expression as follows:

**Conceptual-schema = Conceptual-design(Users-requirements)**

**Logical-schema = Logical-design(Conceptual-schema)**

**Physical-schema = Physical-design(Logical-schema)**

**DDL-code = Coding(Physical-schema)**

Clearly, all these processes are model-driven transformations and can then be described by transformation plans. The level of formality of these processes depends on the methodology, on the availability of a CASE tool, and on nonfunctional requirements, such as performance and robustness, that generally require human expertise. For instance, conceptual design is a highly informal process based on human interpretation of complex information sources, while logical design can be completely described by a transformation plan (such as that of Figure 5 for relational databases). Anyway, these processes can be decomposed into subprocesses, which, in turn, can be modelled by transformations and described by transformation plans, and so forth, until the latter is reduced to elementary operators.

## Transformation-Based Database Reverse Engineering

Modelling database design in this way provides us with an elegant basis to describe database reverse engineering. Indeed the latter can be perceived to be the inverse of database design. An in-depth analysis of reverse engineering can be found in Hainaut (2002).

## CONCLUSION

In this article, we have shown that schema transformations can be used as a major paradigm in database engineering. In particular, being formally defined, it can be used to precisely model complex processes and to reason on their properties such as semantics preservation. It has also been used to derive new processes from former ones, as illustrated by the formalization of database reverse engineering as the inverse of database design.

Due to their formality, transformations can be implemented in CASE tools, either as implicit operators or as tools that are explicitly made available to the developer. Two implementations are worth being mentioned, namely, Rosenthal and Reiner (1994) and Hainaut et al. (1996). The latter reference describes the DB-MAIN CASE environment, which includes a transformation toolbox as well as special engines for user-defined predicate-driven and model-driven transformations. Further information can be found at <http://www.info.fundp.ac.be/libd>.

Transformations also have a great potential in other domains such as database interoperability, in which mediation between existing databases (McBrien & Poulouvasilis, 2003) and data wrapping (Thiran & Hainaut, 2001) can be formalized and automated through transformational operators. In this domain, data instance transformations are modelled by the *t* part of the transformations. Specifying how the source schema is transformed into the target schema automatically provides a chain of instance transformations that are used to generate the data conversion code that is at the core of data migrators (ETL processors), wrappers, and mediators.

## REFERENCES

- Balzer, R. (1981). Transformational implementation: An example. *IEEE TSE*, *SE-7*(1).
- Casanova, M., & Amaral De Sa, A. (1984). Mapping uninterpreted schemes into entity-relationship diagrams: Two applications to conceptual schema design. *IBM Journal of Research & Development*, *28*(1).
- Fikas, S. F. (1985). Automating the transformational development of software. *IEEE TSE*, *SE-11*.
- Hainaut, J.-L. (1995, September). Transformation-based database engineering. VLDB'95 Tutorial notes. University of Zürich, Switzerland. Retrieved from <http://www.info.fundp.ac.be/libd>
- Hainaut, J.-L. (1996). Specification preservation in schema transformations: Application to semantics and statistics. *Data & Knowledge Engineering*, *11*(1).



Hainaut, J.-L. (2002). Introduction to database reverse engineering In *LIBD lecture notes*, University of Namur, Belgium. Retrieved Sept 2004 from <http://www.info.fundp.ac.be/~dbm/publication/2002/DBRE-2002.pdf>

Hainaut, J.-L., Chandelon, M., Tonneau, C., & Joris, M. (1993). Contribution to a theory of database reverse engineering. In *Proceedings of the IEEE Working Conference on Reverse Engineering*, Los Alamitos, CA (pp. 161-170). IEEE Computer Society Press.

Hainaut, J.-L., Englebort, V., Henrard, J., Hick, J.-M., & Roland, D. (1996). Database reverse engineering: From requirements to CASE tools. *Journal of Automated Software Engineering*, 3(1).

Halpin, T. A., & Proper, H. A. (1995). Database schema transformation and optimization. In *Proceedings of the 14th International Conference on ER/OO Modelling (ERA)* (Vol. 1021, pp. 191-203). Berlin/Heidelberg, Germany: Springer LNCS.

Jajodia, S., Ng, P. A., & Springsteel, F. N. (1983). The problem of equivalence for entity-relationship diagrams. *IEEE TSE, SE-9*(5).

Kobayashi, I. (1986). Losslessness and semantic correctness of database schema transformation: Another look of schema equivalence. *Information Systems*, 11(1), 41-59.

McBrien, P., & Poulouvassilis, A. (2003). Data integration by bi-directional schema transformation rules. In *Proceedings of the 19th International Conference on Data Engineering (ICDE'03)*, Los Alamitos, California (pp. 227-238). IEEE Computer Society Press.

Navathe, S. B. (1980). Schema analysis for database restructuring. *ACM TODS*, 5(2).

Omelayenko, B., & Klein, M. (Eds.). (2003). *Knowledge transformation for the Semantic Web*. Amsterdam, The Netherlands: IOS Press.

Partsch, H., & Steinbrüggen, R. (1983). Program transformation systems. *Computing Surveys*, 15(3).

Rauh, O., & Stickel, E. (1995). Standard transformations for the normalization of ER schemata. In *Lecture notes in computer science. Proceedings of the CAiSE95 Conference* (Vol. 932, pp. 313-326). Berlin/Heidelberg, Germany: Springer-Verlag.

Rosenthal, A., & Reiner, D. (1994). Tools and transformations—rigorous and otherwise—for practical database design. *ACM TODS*, 19(2).

Thiran, P., & Hainaut, J.-L. (2001). Wrapper development for legacy data reuse. In *Proceedings of WCRE'01*, Los Alamitos, CA (pp. 198-207). IEEE Computer Society Press.

van Bommel, P. (Ed.). (2004). *Transformation of knowledge, information and data: Theory and applications*. Hershey, PA: Idea Group.

## KEY TERMS

**Inverse Transformation:** Considering schema transformation  $\Sigma$ , the transformation  $\Sigma'$  is the inverse of  $\Sigma$  if its application undoes the result of the application of  $\Sigma$ . The instance mapping of  $\Sigma'$  can be used to undo the effect of applying the instance mapping of  $\Sigma$  on a set of data.

**Model-Driven Transformation:** A goal-oriented chain of transformations made up of predicate-driven operators. It is designed to transform any schema expressed in model M into an equivalent schema in model M'.

**Mutation Transformation:** A schema transformation that changes the nature (entity type, relationship type, attribute) of a schema construct. For instance, an entity type is transformed into a relationship type and conversely.

**Predicate-Driven Transformation:** A couple  $\langle \Sigma, p \rangle$  where  $\Sigma$  is a schema transformation and p a structural predicate that identifies schema patterns. The goal of a predicate-based transformation is to apply  $\Sigma$  to the patterns that meet p in the current schema.

**Schema Transformation:** A transformation is a rewriting rule through which the instances of some pattern of an abstract or concrete specification are replaced with instances of another pattern. The concept also applies to database schemas where the rewriting rule replaces a set of constructs of a database schema with another set of constructs. Such a transformation comprises two parts—a schema rewriting rule (structural mapping) and a data conversion rule (instance mapping). The latter transforms the data according to the source schema into data complying with the target schema.

**Semantics-Preserving Transformation:** A schema transformation that does not change the information contents of the source schema. Both schemas describe the same universe of discourse.

**Transformational Software Engineering:** A view of software engineering through which the production and evolution of software can be modelled, and practically carried out, by a chain of transformations which preserve some essential properties of the source specifications. Program compiling but also transforming tail recursion into an iterative pattern are popular examples. This approach is currently applied to software renovation, reverse engineering, and migration.

# Ubiquitous Computing and Databases

**George Roussos**

*University of London, UK*

**Michael Zoumboulakis**

*University of London, UK*

## INTRODUCTION

The concept of the so-called ubiquitous computing was introduced in the early 1990s as the third wave of computing to follow the eras of the mainframe and the personal computer. Unlike previous technology generations, ubiquitous computing recedes into the background of everyday life:

*It activates the world, makes computers so imbedded, so fitting, so natural, that we use it without even thinking about it, and is invisible, everywhere computing that does not live on a personal device of any sort, but is in the woodwork everywhere.* (Weiser & Brown, 1997, p. 81)

Ubiquitous computing is often referred to using different terms in different contexts. Pervasive, 4G mobile, and sentient computing or ambient intelligence also refer to the same computing paradigm. Several technical developments come together to create this novel type of computing; the main ones are summarized in Table 1 (Davies & Gellersen 2002; Satyanarayanan 2001).

## BACKGROUND

One of the major challenges in turning the ubiquitous computing vision into reality is the development of distributed system architectures that will support effectively and efficiently the ability to instrument the physical world (Estrin et al., 2002; National Research Council, 2001). Such architectures are being developed around two core concepts: self-organizing networks of embedded devices with wireless communication capabilities and data-centricity. To augment physical artifacts with computational and communications capabilities, it is necessary to enable miniaturized hardware components capable of wireless communication. However, these same characteristics that allow for instrumentation of physical objects also impose significant constraints. Systems architectures require significant changes due to the severely limited resources available on these

devices. One possible solution is offered by the emergence of data-centric systems. In this context, data-centric refers to in-network processing and storage, carried out in a decentralized manner (Estrin et al., 2000).

One objective of data-centricity is to let systems exploit the anticipated high node densities to achieve longer unsupervised operating lifetimes. Indeed, smaller form factor wireless sensor nodes have limited resources and often cannot afford to transfer all the collected data to the network edge and forward to centralized information processing systems. A practical example of the data-centric approach for network routing is directed diffusion (Intanagonwiwat, Govindan & Estrin, 2000). This mechanism employs in-network processing by routing data along aggregation paths, thus, removing the need for an address-centric architecture. It exploits data naming as the lowest level of system organization and supports flexible and efficient in-network processing.

In summary, databases have a dual role to play in ubiquitous computing: in the short term, they need to provide the mapping between physical and virtual entities and space in a highly distributed and heterogeneous environment. In the longer term, database management systems need to provide the infrastructure for the development of data-centric systems. Each of the two phases is discussed in turn in the following sections.

## DATABASES IN UBIQUITOUS COMPUTING

As noted earlier, a key requirement of ubiquitous computing is the use of contextual information to adapt system behavior on-the-fly and deliver services that fit the specific user situation, a feature that has become known as context-awareness (Abowd & Mynatt, 2000). Context-aware systems need to combine three elements: details of the physical environment must be sourced from embedded sensors, user profiles must be retrieved from distributed repositories holding fragments of identity information, and these data must be correlated to a domain-specific knowledge model that can be used to



Table 1. Elements of ubiquitous computing

<p><b>1. Physical and Virtual Integration</b></p> <p><b>Sensing</b> Information gathering in the physical world and its representation in the virtual world</p> <p><b>Actuation</b> Decisions in the virtual world and their tangible results in the physical world</p> <p><b>Awareness and Perception</b> Using sensed data to maintain a higher level model of the physical world and argue about it</p> <p><b>Ambient Displays</b> Display information from the virtual world on physical artifacts</p> <p><b>World Modeling</b> Representations of physical spaces</p>
<p><b>2. System Components</b></p> <p><b>Platforms</b> Mobile, wearable or implantable hardware devices with small form factor</p> <p><b>Sensors and Actuators</b> Hardware and software platforms for sensing and actuation</p> <p><b>Software Architectures</b> Adaptable, large scale, complex software infrastructures and development</p> <p><b>Connectivity</b> High-speed, low power wired and wireless communications systems</p>
<p><b>3. Deployment</b></p> <p><b>Scalability</b> System adaptation to cater for massive scale in terms of space, devices, and users</p> <p><b>Reliability</b> Security and redundancy technologies for continuous operation</p> <p><b>Maintenance</b> Structured maintenance processes for reliability and updates</p> <p><b>Evaluation</b> Methods to evaluate the effectiveness of information systems outside the laboratory</p>

reason about the specific situation. In these circumstances, databases must support two aspects of the virtual-physical integration: on the one hand, provide mappings between physical artifacts and associated digital resources and, on the other, hold profile, historical, and state data. This functionality is best illustrated in a case study, for example, ubiquitous retail (Kourouthanassis & Roussos, 2003).

### Database Systems Challenges

The introduction of ubiquitous computing technologies in a retail environment can help develop a number of novel applications that can transform the shopping experience and at the same time create value for retailers. Some of the possibilities that appear attractive to consumers include the smart shopping cart, in-store navigational assistance, shelf displays, and offers and promotions targeted to the individual. At the same time, retailers can use these applications to gather data to help optimize the supply chain and increase profit margins, while at the

same time improving consumer satisfaction. A core ingredient for such ubiquitous retail systems is the augmentation of consumer products with information processing capabilities and the development of supporting infrastructures.

To this end, several new technologies have been recently introduced. The Electronic Product Code (EPC) is a globally unique identifier that is attached to a particular product item stored in a radio frequency identification (RFID) tag. Unlike barcodes that are used to identify product classes, the EPC distinguishes a particular product instance, for example a specific can of cola rather the class of cola cans, and may be used to trace additional information about it. This information is retrieved by querying the Object Naming Service (ONS), which matches the EPC to the production server that holds information about the item in Product Markup Language (PML), for example, its ingredients and date of production (Mealling, 2003). Using these technologies, a smart shopping cart will identify the products placed into it using the EPC, retrieve additional information about the

product via ONS and PML, and provide personalized feedback to the consumer. For example, product information can be checked against the user profile retrieved from the consumer's identity management service provider for specific medical counter indications, for example, lactose intolerance.

This example highlights some of the immediate challenges of ubiquitous computing for database technology. Firstly, database management must address the demands of a highly heterogeneous environment, both from a technical and an organizational point of view. Indeed, data resources are highly geographically distributed, and different service providers control access to specific segments of the data space. Moreover, data generation and query insertion may move rapidly from one location to another to the extent that almost every object in the system is transient. Finally, these requirements put novel demands on system architectures, and we will discuss each of these issues in turn in the remainder of this section.

Ubiquitous computing databases need to cater for the physical distribution of data production, data consumption, and data storage over wide geographic regions. Indeed, geographic distribution affects both data sources and applications, which are mobile to a very high degree. In the ubiquitous retail scenario introduced previously, a product item, for example, a single can of cola, is a data source that may travel thousands of miles from its production plant until consumed and discarded. In the same scenario, consumers would wish to access personalized information services irrespective of their location in the world.

In addition to geographic distribution, ubiquitous computing infrastructures are also logically distributed since they integrate multiple independently controlled components and subsystems. It is not rare that one subsystem may be in operation in some form for decades but would still have to interoperate with newly developed technologies. For example, the shopping cart application discussed earlier retrieves data from the user's identity management service provider, the ONS system, the manufacturer PML servers, and the retailer databases and still has to perform without significant delays for the user while also providing up-to-date information. This data source heterogeneity demands that management and acceleration architectures must adapt to accommodate application requirements.

Further, the origin of a particular query may also have considerable impact on the organization and performance of a distributed query-processing architecture. To be sure, distribution of queries and the rate at which they are initiated will affect system performance. For example, reading the EPC of a new item placed inside a smart shopping cart will generate a new query to the PML server at the rate at which the consumer adds new items in the cart. To improve performance, some systems may

correlate data distribution with query distribution to control their overhead. Alternatively, systems may restrict the set of available queries and thus limit their impact on system throughput. One solution to this problem is the construction lower-cost overlay networks for processing queries. Such overlays, currently referred to as service-oriented architectures, allow for data sources and sinks to connect to the system via the closest access point.

Ubiquitous computing devices will also typically communicate over unreliable networks. Thus, failure should be a feature of the system and should be taken into account in designing appropriate query-processing architectures. For example, all objects in the database may be persistent so that when a client reconnects to the network, the system may replicate the object closer to the new location of the client and thus reduce latency. This feature would also support correlation of application queries and the data they carry, as is the case of a client that initiates a specific query for data but also becomes a source for the data after the query is satisfied.

Finally, developing optimized architectures for ubiquitous computing systems requires a better understanding of typical workload patterns. Consider the case of the smart shopping cart, which generates data and queries for data. In a realistic scenario of a supermarket chain, the pattern of the generated workload is rather unique and would induce considerable stress on existing database system architectures primarily due to the high utilization of scarce resources. Because of the limited experience we have with realistic workloads, it may be inappropriate to optimize systems architectures. At the same time, the stress that ubiquitous computing systems will put on public resources, for example, the ONS, will demand that massively distributed infrastructures are developed at a scale orders of magnitude larger to that available today.

## Sensor Databases

A particularly important development for databases in ubiquitous computing is the emergence of the so-called sensor and actuator databases. Recent advances in micro-electro-mechanical systems (MEMS) and wireless communication, coupled with significant improvements in electronics manufacturing processes have enabled the development of low-cost, low-power, multi-functional sensor and actuator nodes (Akyildiz et al., 2002). Such nodes are small in size and can communicate untethered in relatively short distances, and thus it has become practical for these devices to be embedded into an increasing range of physical artifacts. Moreover, sensor and actuator nodes can be linked together in dense, self-configurable, and adaptively co-



ordinated networks, often called sensor and actuator networks. These systems are also referred to as Smart Dust, wireless networks of sensors and actuators, motes, embedded wireless networks, or smart-its. The name Smart Dust seems to have particular appeal with the term *dust* justified on the basis that many of these nodes are expected to be less than one cubic millimeter in volume (currently 2.5mm<sup>3</sup> sensors are in production, with 0.15mm<sup>3</sup> nodes in laboratory testing).

Because individual nodes are embedded in objects, sensor and actuator networks are tightly coupled to the physical world. In this sense, they represent a realization of ubiquitous computing's Holy Grail to provide a "connection of things in the world with computation" (Weiser, 1991). Although the same goal can be achieved using alternative technologies, sensors and actuators offer a competitive candidate for the deployment of ubiquitous, self-organizing networks of artifacts. Indeed, their cost and performance characteristics combined with their infrastructure-free operation, and their low or no dependence on supporting computational and communications fabric offers distinct advantages in deployment and operation.

Yet, such flexibility does not come without constraints. Sensor and actuator nodes are also highly resource constrained and often create massively concurrent, complex, networked, computational systems. Moreover, they are integrated into objects and systems that are likely to function for long periods of time unsupervised and used by individuals who are not technology experts. Finally, they assemble information processing and communication systems with significantly greater size and scale compared to those that exist today.

Sensor and actuator networks operate unsupervised with individual nodes making decisions independently in a decentralized manner. Applications frequently operate in one of either modes:

1. In event-driven applications, for example, detection of forest fires, the system remains inactive until an event is generated in one of the nodes. Then, the event propagates through the system and causes the activation of appropriate behavior.
2. In demand-driven applications, for example, inventory tracking, activity is initiated in response to external requests, usually in the form of queries.

Both cases include a data transmission step, which may be either proactive or reactive. That is, data transmission may be in response to a particular query inserted to the network by an external device or it may be triggered by a condition that is checked at a particular node subset inside the network itself. Furthermore, in both cases,

system behavior is defined in terms of high-level statements of logical interests over the entire network, and they might be represented either in an appropriate query definition language (Gehrke & Madden, 2004) or as event-condition-action triplets (Zoumboulakis, Roussos & Poulouvasilis, 2004).

In the traditional embedded software development model, each node would be treated as a separate computational unit and programmed in a C-derived language. This model cannot be effectively extended to sensor and actuator networks where there is a clear need of a higher-level language to define system behavior in terms of a more amenable programming framework. To this end, query processing has attracted considerable interest and is rapidly becoming a popular computational paradigm for a plethora of sensor network applications. This approach has been seen to address well the complex requirements of application development in sensor networks in a variety of applications including environmental monitoring, distributed mapping, and vehicle tracking. Prototype sensor network query processors have been implemented in Tiny DB (Madden et al., 2003) and Cougar (Yao & Gehrke, 2002) systems. A second database technology that provides an appropriate computational model for event-based sensor and actuator network applications is event-condition-action (ECA) rules (Zoumboulakis, Roussos & Poulouvasilis, 2004).

Both technologies viewed as a computational paradigm for sensor and actuator networks are successful in freeing developers from lower level concerns by transferring responsibility of data management, collection, and processing to the system (Bonnet, Gehrke & Seshadri, 2001). Despite their early successes, there are still several challenges associated with the design and development of such sensor databases:

- **Node resource management:** Sensor and actuator nodes are resource-constrained, and any system should carefully manage their consumption, especially power. Communication and sensing tend to dominate demands on the battery while computation places a relatively lower burden.
- **Network topology management:** Sensor and actuator networks have a fundamentally transient nature due to moving nodes, signal degradation, changing interference patterns, and nodes running out of power.
- **Data aggregation:** Due to the high cost of communication and the possible very high data collection rates, it is desirable to summarize or abstract the recorded data as they propagate through the network. For example, decomposable operators may be



computer in the fly by intermediate nodes and only the aggregates forwarded.

- **Simple and expressive language:** To preserve node resources, the query or the ECA language used to interact with the sensor network should be simple enough to support a lightweight implementation and expressive enough to be useful in building applications.
- **Management tools:** In addition to an appropriate conceptual framework, a system becomes useful when it also offers a collection of tools to assist users in management and interaction.

One implication of these requirements is that it is often necessary for the nodes to perform collaborative signal processing to effectively analyze the sensed data (Estrin, Govidan et al., 1999). Furthermore, the high probability of node failures implies that sensor data will be noisy, and thus managing sensor data uncertainty should be adequately addressed. Another critical concern is the designing of efficient routing mechanisms that cater for the particular requirements of query processing and reactive functionality. As a result, it is often necessary to develop database systems that bypass the traditionally distinct network layers.

A further implication is that query dissemination or corresponding construction of an event channel for ECA rule execution often requires two phases. First is the preprocessing of a routing tree that defines the pathways along which data travels. This routing tree is constructed online as nodes forward notifications to their selected neighbors in the network. When signals are generated at sensor nodes, the recorded data is possibly aggregated with information received by other sensors and subsequently forward to parent nodes until data reaches its intended destination. The details of such routing algorithms are beyond the scope of this article, but the interested reader may consult Gherke and Madden (2004) for full details.

## FUTURE TRENDS

In the next decade, databases employed in support of ubiquitous computing systems will be extended to cater for geographically decentralized operation, both in terms of data generation and application access, the transient nature of ubiquitous computing objects and entities, the fundamentally heterogeneous nature of such a system, and the need for correlation and coordination of physical and virtual resources. New systems architectures will be developed to support ubiquitous computing systems and the novel workloads they bring with them.

A particularly interesting ubiquitous computing system that is expected to attract considerable interest over the next years is sensor databases. Sensor databases consist of numerous tiny embedded computing devices capable of wireless communication. Such systems have extremely limited resources and must take advantage of in-network processing and storage to survive for longer periods of time. Database management systems for sensor network offer distinct advantages from a user interaction perspective but also face considerable challenges for efficient system operation.

## CONCLUSION

Ubiquitous computing offers the promise of pervasive information and communications infrastructures that provide their services whenever required while taking into account the particular context of the user. Ubiquitous computing also blurs the boundaries between the physical and the virtual worlds through sensing and actuation technologies. To realize this vision, developments in several technical areas are required, and database management systems have a critical role to play in supporting ubiquitous computing services.

## REFERENCES

- Abowd, G., & Mynatt, E. (2000). Charting past, present, and future research in ubiquitous computing. *ACM Transactions on Computer-Human Interaction*, 7(1), 29-58.
- Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002). Wireless sensor networks: A survey. *Computer Networks*, 38, 393-422.
- Bonnet, P., Gehrke, J.E., & Seshadri, P. (2001). Towards sensor database systems. *Proceedings of the 2nd International Conference on Mobile Data Management*, Hong Kong, China, January.
- Davies, N., & Gellersen, H. (2002). Beyond prototypes: Challenges in deploying ubiquitous systems. *IEEE Pervasive Computing*, 1(1), 29-35.
- Estrin, D., Culler, D., Pister, K., & Sukhatme, G. (2002). Connecting the physical world with pervasive networks. *IEEE Pervasive Computing*, 1(1), 59-69.
- Estrin, D., Govindan, R., Heidemann, J., & Kumar, S. (1999). Next century challenges: Scalable coordination in sensor networks. *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking* (pp. 263-270). Seattle, Washington.

Gehrke, J., & Madden, S. (2004). Query processing in sensor networks. *IEEE Pervasive Computing*, 3(1), 46-55.

Intanagonwiwat, C., Govindan, R., & Estrin, D. (2000). Directed diffusion: A scalable and robust communication paradigm for sensor networks. *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCOM'00)*, Boston, August.

Kourouthanassis, P., & Roussos, G. (2003). Developing consumer-friendly pervasive retail systems. *IEEE Pervasive Computing*, 2(2), 32-39.

Madden, S., Franklin, M.J., Hellerstein, J.M., & Hong, W. (2003). The design of an acquisitional query processor for sensor networks. *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data* (pp. 491-502), San Diego, California.

Mealling, M. (2003). *Object name service (ONS) 1.0*. Auto-ID Center, Boston.

National Research Council. (2001). *Embedded, everywhere: A research agenda for networked systems of embedded computers*. National Academy Press, Washington DC, USA.

Satyanarayanan, M. (2001). Pervasive computing: Vision and challenges. *IEEE Personal Communications*, 8(4), 10-17.

Weiser, M. & Brown, J.S. (1997). The coing age of calm technology. In P. Denning, R.M. Metcalf (Eds.), *Beyond calculation* (pp. 75-85).

Yao, Y., & Gehrke, J.E. (2002). The cougar approach to in-network query processing in sensor networks. *Sigmod Record*, 31(3).

Zoumboulakis, M., Roussos, G., & Poulouvasilis, A. (2004). *Asene: Active sensor networks*. *Proceedings of the International Workshop on Data Management for Sensor Networks (VLDB 2004)*, Toronto, Canada, August 30.

## KEY TERMS

**Data Aggregation:** The process in which information is gathered and expressed in a summary form. In case the aggregation operator is decomposable, partial aggregation schemes may be employed in which intermediate results are produced that contain sufficient information to compute the final results. If the aggregation operator is non-decomposable, then partial aggregation schemes can still be useful to provide approximate summaries.

**In-Network Processing:** A technique employed in sensor database systems whereby the data recorded is processed by the sensor nodes themselves. This is in contrast to the standard approach, which demands that data is routed to a so-called sink computer located outside the sensor network for processing. In-network processing is critical for sensor nodes because they are highly resource constrained, in particular, in terms of battery power, and this approach can extend their useful life quite considerably.

**Sensor and Actuator Networks:** An ad-hoc, mobile, wireless network consisting of nodes of very small size either embedded in objects or freestanding. Some nodes have sensing capabilities, which frequently include environmental conditions, existence of specific chemicals, motion characteristics, and location. Other nodes have actuation capabilities; for example, they may provide local control of mechanical, electrical, or biological actuators and optical long-range communications.

**Sensor Databases:** The total of stored sensor data in a sensor and actuator networked is called a sensor database. The data contain metadata about the nodes themselves, for example, a list of nodes and their related attributes, such as their location, and sensed data. The types and the quantity of sensed data depends on the particular types of sensors that participate in the network.

**Sensor Query Processing:** Sensor query processing is the design of algorithms and their implementations used to run queries over sensor databases. Due to the limited resources of sensor and actuator nodes query processing must employ in-network processing and storage mechanisms.

**Service-Oriented Architectures:** A service is an information technology function that is well defined, self-contained, and does not depend on the context or state of other services. A service-oriented architecture is an approach to building information technology systems as a collection of services which communicate with each other. The communication may involve either simple data passing between services, or it could involve infrastructure services which coordinate service interaction. SOA is seen as a core component of ubiquitous computing infrastructures.

**Ubiquitous Computing:** A vision of the future and a collection of technologies where information technology becomes pervasive, embedded in the everyday environments and thus invisible to the users. According to this vision, everyday environments will be saturated by computation and wireless communication capacity, and yet they would be gracefully integrated with human users.

# Using Semantic Web Tools for Ontologies Construction

Gian Piero Zarri

University of Paris IV/Sorbonne, France

## INTRODUCTION: ONTOLOGIES AND SEMANTIC WEB

The current state of Web technology—the “first generation” or “syntactic” Web—gives rise to well-known serious problems when trying to accomplish, in a non-trivial way, essential tasks like indexing, searching, extracting, maintaining, and generating information. These tasks would, in fact, require some sort of “deep understanding” of the information dealt with. In a “syntactic” Web context, on the contrary, computers are only used as tools for posting and rendering information by brute force. Faced with this situation, Tim Berners-Lee first proposed a sort of “Semantic Web,” where the access to information is based mainly on the processing of the *semantic properties* of this information: “The Semantic Web is an extension of the current Web in which information is given well-defined *meaning* [italics added], better enabling computers and people to work in cooperation” (Berners-Lee, Hendler, & Lassila, 2001, p. 35). The Semantic Web’s challenge consists then in being able to access and retrieve information on the Web by “understanding” its proper semantic content (its meaning) and not simply by matching some keywords.

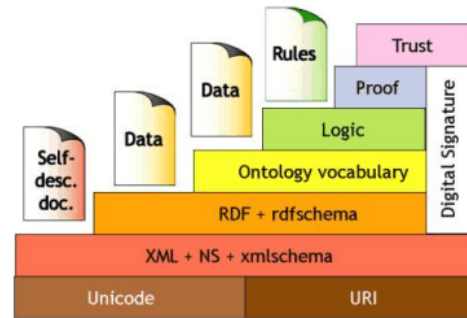
From a technical point of view, the Semantic Web vision is deeply rooted into an “ontological” approach, with some proper characteristics that differentiate it from the “classical” approach to the construction of ontologies that has been described in companion article of this encyclopedia title “Ontologies and Their Practical Implementation.” We will describe these characteristics in the following sections.

## BACKGROUND INFORMATION

### Berners-Lee’s Architectural Proposal for the Semantic Web

To support his vision, Berners-Lee has proposed in several talks an “architecture” for the Semantic Web like that reproduced in Figure 1. Making abstraction now from all the discussions and criticisms that this

Figure 1. Semantic Web architecture according to Tim Berners-Lee



proposal has brought up, what is relevant for the topic of this article is the central position that “ontologies” occupy in the architecture—and that nobody wishes to challenge. A first important difference with respect to what is expounded in the article “Ontologies and Their Practical Implementation” is, however, that ontologies are no more considered “in isolation”: They are now supported by lower-level tools like XML and RDF and must also implement an additional logic level.

In the embedded architecture of Figure 1, Unicode and URI make up the basis of the hierarchy. The Unicode standard provides a unique numerical code for every character that can be found in documents produced according to any possible language, no matter what are the hardware and software used to deal with such documents. It is supported in many operating systems and all the modern browsers, and it enables a single software product or a single Web site to be targeted across multiple platforms, languages, and countries without reengineering. URI (Uniform Resource Identifier) represents a generalization of the well-known URL (Uniform Resource Locator), which is used to identify a “Web resource” (e.g., a particular page) by denoting its primary access mechanism (essentially, its “location” on the network). URI has been created to allow recording information about all those “notions” that, unlike Web pages, do not have network locations or URLs but that need to be referred to in an RDF statement. These notions include network-accessible things, such as an electronic document or an image, and things that are not

network-accessible, such as human beings, corporations, and bound books in a library, or abstract concepts like the concept of a “creator.”

XML (Extensible Markup Language; see Bray, Paoli, Sperberg-McQueen, Maler, & Yergeau, 2004), has been created to overcome some difficulties proper to HTML (Hypertext Markup Language), developed in 1989 by Tim Berners-Lee as a means for sharing information from any location. An HTML file is a text file characterized by the presence of a small set of “tags”—like `<Head>`, `<Body>`, `<Input>`, `<Applet>`, `<Font>`, etc.—that instruct the Web browsers how to display a given Web page. HTML is, then, a “presentation-oriented” markup tool. In spite of its evident utility, HTML suffers from a number of limitations, from its lack of efficiency in handling the complex client/server communication of today’s applications to (mainly) the impossibility of defining new tags to customize exactly the user’s needs. XML is called “extensible” because, at the difference of HTML, it is not characterized by a fixed format but lets the user design his own customized markup languages (a specific DTD, Document Type Description, see below) for limitless different types of documents; XML is a “content-oriented” markup tool. Basically, the syntactic structure of XML is very simple. Its markup elements are normally identified by an opening and a closing tag, like `<employees>` and `</employees>` and may contain other elements or text. The elements must be properly nested, and every XML document must have exactly one root element. Markup elements can be characterized by adding attribute/value pairs inside the opening tag of the element, like `<person name=“Mary”>`. Taking into account the nesting constraint, a very simple fragment of XML document could then be represented as: `<employees> <person name=“Mary”> <id>99276</id> </person> </employees>`. To allow a computer to interpret correctly a fragment like this, it is necessary, however, to specify the semantics of the markup elements and tags used to make up it; a simple way of doing this is to make use of a DTD. A DTD is a formal description in XML declaration syntax of a particular type of document. It begins with a `<!DOCTYPE` keyword and sets out what names are to be used for the different types of markup elements, where they may occur, the elements’ possible attributes, and how they all fit together. For example, a DTD may specify that every **person** markup element must have a **name** attribute, and that it can have an offspring element called **id** whose content must be text. Before reading an XML document, the validating parsers and the application programs (editors, search engines, navigators, databases) read the corresponding DTD so that they can identify where every element type ought to come and how each relates to the other. There are many

sorts of DTDs ready to be used in all kinds of areas (see, e.g., <http://www.w3.org/QA/2002/04/valid-dtd-list.html#full>) that can be downloaded and used freely. Some of them are MathML, for mathematical expressions; SMIL, Sync Multimedia Integration Language; CML, Chemical Markup Language; OSD, Open Software Description; EDI, Electronic Data Interchange; PICS, Platform for Internet Content Selection; etc. A more complete way of specifying the semantics of a set of XML markup elements is to make use of XML Schema (as mentioned in Figure 1). XML Schema (see Biron & Malhotra, 2001; Thompson, Beech, Maloney, & Mendelsohn, 2001) supplies a more complete grammar for specifying the structure of the elements allowing one, e.g., to define the cardinality of the offspring elements, default values, etc.

## RDF, Resource Description Framework

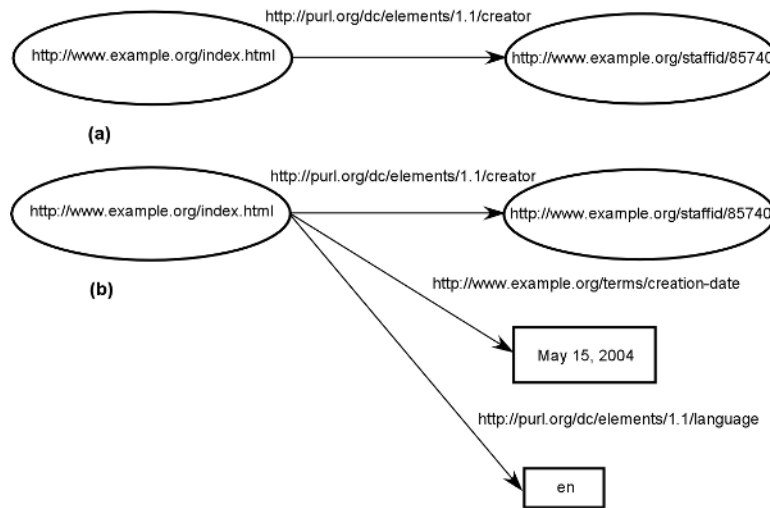
Moving up in the structure of Figure 1, we find now RDF (Resource Description Framework), an example of “metadata” language (metadata = data about data) used to describe generic “things” (“resources,” according to the RDF jargon) on the Web. An RDF document is, basically, a list of statements under the form of triples having the classical format: `<object, property, value>`, where the elements of the triples can be URIs (Universal Resource Identifiers, see above), literals (mainly, free text), and variables. To follow a well-known RDF example—reproduced, e.g., in the 2004 edition of the “official” *W3C RDF Primer* (Manola & Miller)—let us suppose we want to represent a situation where someone named John Smith has created a particular Web page. We will then make use of the RDF triple: `<http://www.example.org/index.html (object), creator (property), john_smith (value)>`. Adding additional information about the situation, by stating, e.g., that the Web page was created May 15, 2004, and that the language in which the page is written is English, amounts to adding two statements: `<http://www.example.org/index.html (object), creation_date (property), May 15, 2004 (value)>` and `<http://www.example.org/index.html (object), language (property), English (value)>`. Note that, unfortunately, RDF uses a particular terminology for denoting the three elements of the triples, calling then subject, predicate, and object, respectively, the object, property, and value elements of the triples. This decision is unfortunate because it introduces an undue confusion with well-defined and totally different linguistic categories.

RDF triples are very easily represented as directed labeled graphs, by denoting resources as ovals, properties (predicates) as arrows, and literal values like **May 15, 2004** or **English** within boxes. Figure 2a represents then in graph form the original statement: “John Smith has





Figure 2. RDF statements represented into graph format



created a Web page.” The addition of information about date and language gives rise to the graph of Figure 2b, given that groups of statements are represented by corresponding groups of nodes and arcs. Note that, to simulate the actual conditions of utilization of RDF, the properties **creator**, **creation\_date**, and **language** in Figure 2 have been replaced, respectively, by `http://purl.org/dc/elements/1.1/creator`, `http://www.example.org/terms/creation-date`, and `http://purl.org/dc/elements/1.1/language`; analogously, **john\_smith** has been replaced by `http://www.example.org/staffid/85740`. All these “`http://...`” terms are URIs that identify in an unambiguous way specific RDF entities; more exactly, they refer to the ontologies/metadata repositories/lists of reserved domain names where these entities are defined. For example, “`http://purl.org/dc/...`” refers to the collection of metadata terms maintained by the Dublin Core Metadata Initiative (see, e.g., Dekkers & Weibel, 2003). In this collection, e.g., `http://purl.org/dc/elements/1.1/creator` is defined as: “An entity primarily responsible for making the content of the

resource.” The literal **en** (Unicode characters) is an international standard two-letter code for English, see `http://purl.org/dc/elements/1.1/language`; the **example.org** Internet domain name is reserved for documentation purposes.

From what has been expounded until now, RDF seems to be nothing more than a downgraded form, Internet-oriented, of semantic networks as they were used in the artificial intelligence domain at the beginning of the 1970s. Its significance in a Semantic Web context becomes more evident when we examine the way of writing RDF statements into XML format—the so-called RDF/XML syntax (see Beckett, 2004)—i.e., when RDF is seen as a sort of additional DTD of XML. Table 1 reproduces then the simple example of Figure 2b, making use of the RDF/XML syntax.

The first line of the code, `<?xml version="1.0"?>` is the “XML declaration,” which states that what follows consists of XML and which specifies the version used. In the second line we find an XML markup element that

Table 1. The RDF/XML syntax

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:exterm="http://www.example.org/terms/">
  <rdf:Description rdf:about="http://www.example.org/index.html">
    <exterm:creation-date>May 15, 2004</exterm:creation-date>
    <dc:language>en</dc:language>
    <dc:creator rdf:resource="http://www.example.org/staffid/85740"/>
  </rdf:Description>
</rdf:RDF>
```



starts with the tag `<rdf:RDF>`—this tag specifies that all the following XML code, until the `</rdf:RDF>` tag of the last line, is intended to represent RDF statements—and ends with the “>” symbol at the right limit of line 4. Within this markup element we find three “XML attributes,” see above, of the opening `<rdf:RDF>` tag; all these attributes (xmlns attributes) have as values the declarations of the namespaces to be used within the RDF/XML code. An attribute like `xmlns:rdf` means that, according to “value” associated with this attribute (after the “=” symbol), all the terms/tags included in this RDF/XML content and prefixed with `rdf:` are part of the namespace identified with the URI: `http://www.w3.org/1999/02/22-rdf-syntax-ns#`; analogously for the `xmlns:dc` (Dublin Core terms) and `xmlns:exterms` (example terms) attributes.

After these preliminary “housekeeping” declarations, lines 5 through 9 represent the core of the RDF/XML coding of the example. The `rdf:Description` start tag of line 5 indicates that we are now introducing the “description” of a resource; this resource, `http://www.example.org/index.html`, is identified as the value of the `rdf:about` attribute of the start tag. The three following lines, 6 through 8, are examples of use of “property element” constructions. In these lines, the tags are built up according to the XML QName (Qname = qualified name) convention, which allows shortening the writing of full RDF triples by introducing abbreviations for the URI references. A QName tag contains, in fact, a “prefix” that denotes a given namespace (e.g., `exterms` in line 6) followed, after a colon, by a “local name” (creation-date, i.e., the name of the property). A full URI reference is then created by appending the local name to the URI of the namespace identified by the first part of the QName. For lines 6 through 8, the full URIs become then `http://www.example.org/terms/creation-date`, `http://pur1.org/dc/elements/1.1/Language`, and `http://pur1.org/dc/elements/1.1/creator`. Note that, for the properties corresponding to literals (lines 6 and 7), the values of the properties are directly included within opening and closing QName tags. For the property of line 8, which corresponds to a resource, the value corresponds to the value of the `rdf:resource` attribute of the `dc:creator` QName tag. The description of the resource introduced in line 5 ends with the closing tag of line 9.

To conclude about RDF, we will note that RDF Schema (or RDFS; see Brickley & Guha, 2004) provides a mechanism for constructing specialized RDF vocabularies through the description of domain-specific properties. This is obtained mainly by describing the properties in terms of the classes of resources to which they apply. For example, we could define the `creator` property, saying that it has the resource `document` as “domain” (`document` is the value or “object” of this property) and the resource `person` as “range” (this property must always be associ-

ated with a resource `person`, its “subject”). Other basic modeling primitives of RDFS allow setting up hierarchies (taxonomies), both *hierarchies of concepts*, thanks to the use of “class” and “subclass-of” statements, and *hierarchies of properties*, thanks to the use of “property” and “subproperty-of” statements. Instances of a specific class (concept) can be declared, making use of the “type” statement. Note eventually that, notwithstanding the introduction of these primitives, RDFS is still quite simple compared to “true” knowledge representation languages.

## OWL, The Web Ontology Language

Passing to the next stage of the structure of Figure 1, we can make three general remarks about the notion of “ontology” in a Semantic Web context:

- The basic “nature” of ontologies as described in the article “Ontologies and Their Practical Implementation” does not change fundamentally in this new context: They are still formed by hierarchies (DAGs) of concepts defined through properties and values.
- For their practical implementation, however, these Semantic Web ontologies make large use of the RDF/XML syntactic/semantic constructs.
- Taking also into account that the level that follows “ontology vocabulary” in the pyramid of Figure 1 is “logic,” the Semantic Web ontologies evidence a very strong logic influence.

On February 10, 2004, the W3C published an official recommendation concerning OWL, the Web Ontology Language (see Bechhofer et al., 2004). W3C—the World Wide Web Consortium, coordinated by MIT (USA), ERCIM, the European Research Consortium for Informatics and Mathematics, and the Keio University (Japan)—includes all the main bodies on Earth interested in the developments of Internet and the Web. At the beginning of this document it is stated that: “The Web Ontology Language (OWL) is a semantic markup language for publishing and sharing ontologies on the World Wide Web. OWL is developed as a vocabulary extension of RDF (the Resource Description Framework) and is derived from the DAML+OIL Web Ontology Language. ... An OWL ontology is an RDF graph, which is in turn a set of RDF triples.” The mention of DAML+OIL (McGuinness, Fikes, Hendler, & Stein, 2002) explains the strong logic orientation of OWL—given that OIL (Ontology Inference Layer), the “European” component of DAML+OIL (DAML is the Darpa Agent Markup Language), was implemented in description Logics (DL) terms—DL (Baader, Calvanese, McGuinness, Nardi, & Patel-Schneider, 2002) have



been created to offer a formal foundation for frame-based systems.

OWL consists of three subsets (three specific sublanguages) characterized by an increasing level of complexity and expressiveness: OWL Lite, OWL DL (DL stands for description logics), and OWL Full.

OWL Lite includes only a reduced subset of the OWL language constructors and has a lower formal complexity than the other OWL versions. It is meant mainly to allow (1) the implementation of simple classification hierarchies and (2) the familiarization with the OWL approach. It employs all the features already introduced by RDFS, making use of the same tags—like **rdfs:subclassOf**, **rdfs:subPropertyOf**, **rdfs:domain**, **rdfs:range**—with the same semantics. Note that **rdfs:subclassOf** is the fundamental constructor that is used to set up taxonomies/ontologies in OWL. It relates, in fact, a more specific class (concept) to a more general one: If *X* is a subclass of *Y*, then every instance of *X* is also an instance of *Y*. The relation **rdfs:subclassOf** is transitive: If *X* is a subclass of *Y* and *Y* is a subclass of *Z*, then *X* is a subclass of *Z*. With respect to RDFS, OWL Lite includes several new features:

- Constructors for equality and inequality, i.e., **owl:equivalentClass**, **owl:equivalentProperty**, **owl:sameAs** (two individuals may be stated to be the same), **owl:differentFrom**, **owl:AllDifferent**.
- Constructors used to provide specific information about properties and their values, like **owl:inverseOf**—e.g., stating that the property **hasChild** is the inverse of the property **hasParent**, and stating that Mary is endowed with the property (**hasParent** Lucy) allows then an OWL reasoner to deduce that Lucy is endowed with the property (**hasChild** Mary)—**owl:TransitiveProperty**, and **owl:SymmetricProperty**.
- Constructors used to impose constraints on the way properties can be used by the instances of a class (concept). They are **owl:allValuesFrom** and **owl:someValuesFrom**. For example, **owl:allValuesFrom** introduces a range restriction, imposing, e.g., that the property **hasDaughter** of the class **Person** is restricted to obtaining all its values (**allValuesFrom**) from the class **Woman**. This allows a reasoner to deduce that, if an individual Lucy is related by the property **hasDaughter** with the individual Mary, Mary must be an instance of the class **Woman**.
- Constructors, e.g., **owl:minCardinality** and **owl:maxCardinality**, used to introduce a limited form of cardinality restrictions, stated on the properties of a particular class and to be intended as

constraints on the cardinality of that property when used in the instances of that class. Note that, for algorithmic efficiency reasons, OWL Lite allows using only the integers 0 and 1 to express the cardinality constraints; this restriction is removed in OWL DL.

- OWL Lite includes a (restricted form) of intersection constructor, **owl:intersectionOf**, allowing one, e.g., to state that the class **EmployedPerson** is the **intersectionOf** the classes **Person** and **EmployedThings**.

OWL DL makes use of the full set of the OWL constructors, but it introduces also some constraints on their use to give rise to systems that are “complete” (all the possible deductions are computable) and “decidable” (all the computations will be executed in finite time). Mainly, OWL DL implements what is called “type separation,” which means that a class (concept) cannot be also an individual or a property, and that a property cannot be also an individual or a class. This restriction is removed in OWL Full. OWL DL adds to the OWL Lite list of constructors some new constructors like **owl:oneOf** (classes may be described by enumeration of the individuals that make up the class), **owl:hasValue** (a property is required to have a given individual as value), **owl:disjointWith** (classes can be described as disjoint from each other, see the classes **Man** and **Woman**), **owl:unionOf**, **owl:complementOf**, **owl:intersectionOf** (Boolean combinations of classes), etc.

OWL Full is similar to OWL DL, but for this sublanguage all the constraints have been suppressed—e.g., a class (concept) can be simultaneously treated as a collection of instances (individuals) and as an individual in itself. This can lead to the implementation of systems that are, at least partly, “incomplete” and/or “undecidable.” Currently, no complete implementation of OWL Full exists.

To give now an at least a partial picture of the representation of an ontology in OWL format, we reproduce in Table 2 a small fragment of the OWL version of the “wine” ontology, an ontology often used for exemplification’s purposes in the Semantic Web milieu (see McGuinness et al., 2002; Smith, Welty, & McGuinness, 2004). The code in this table can be considered indifferently as OWL Lite, OWL DL, or OWL Full; in Table 3, we reproduce, on the contrary, another fragment of the wine ontology that makes use of constructors proper to the DL version of the OWL language. Note that, for simplicity’s sake, we have not reproduced in Tables 2 and 3 the “housekeeping” declarations (see Table 1) that are necessary to identify all the XML namespaces associated with the wine ontology: E.g., `xmlns:owl="http://www.w3.org/2002/07/owl#"` is the conventional OWL

declaration that is used to introduce the OWL vocabulary; `xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"`  identifies the elements prefixed as `rdf:` as referring to the RDF namespace, etc.

In the first line of Table 2, the class `Wine` is introduced, making use of an `rdf:ID` attribute. At the difference of the `rdf:about` attribute used in Table 1, `rdf:ID` introduces as its value only a “fragment identifier” (here `wine`) that represents an abbreviation of the complete reference to the URI of the resource being described. The full URI reference is formed by taking the base URI of the wine ontology, e.g., `http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine`, and appending the character `#` (to indicate that what follows is a fragment identifier) and then `wine` to it, giving then the absolute URI reference: `http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#wine`. Note that the `Wine` class can now be referred to by using `#wine`; e.g., `rdf:resource="#wine"` is a well-formed OWL statement. As already stated, the fundamental taxonomic constructor is `rdfs:subClassOf`; the second line of the code of Table 2 allows then the insertion of the class `Wine` into the global ontology by asserting that it is a specialization of the class (concept) `PotableLiquid` (liquid suitable for drinking), which can be defined, in turn, as a specialization of the class `ConsumableThing`.

The third line of the code warns that the class `Wine` is also a specialization of a second class: This last is an

“anonymous” class, whose definition is included within the opening `owl:Restriction` markup element in line 4 and ends with the closing `/owl:Restriction` markup element in line 7. In OWL, in fact, a property restriction on a class is a special kind of class description, that of the anonymous class including all the individuals that satisfy the given restriction. In line 5, the `owl:onProperty` constructor introduces then the name of the property, `madeFromGrape`, to associate with the class `Wine`; line 6 specifies that the cardinality of this property is 1. The insertion of this restriction in the definition of the class `Wine` states, globally, that every specific wine must also be characterized by at least one `madeFromGrape` relation. Note that (1) the `xsd:nonNegativeInteger` data type used to introduce the literal 1 in the `owl:minCardinality` restriction of line 6 is part of the built-in XML Schema data types (see Biron & Malhotra, 2001) and their use is strongly recommended in an OWL context; and (2) the value 1 conforms to the OWL Lite restrictions.

The code fragment of Table 3 defines the class `RedWine` as the precise intersection (logical conjunction “and”) of the class `Wine` and the set of things that are red in color (anonymous class). The presence of the attribute `rdf:parseType="Collection"` is mandatory for this type of construction. Note the use of the DL constructor `owl:hasValue` to impose the value “Red” on the property `hasColor` of the anonymous class.

Table 2. A fragment of the OWL wine ontology

```
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="#PotableLiquid"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#madeFromGrape"/>
      <owl:minCardinality rdf:datatype="xsd:nonNegativeInteger">1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
</owl:Class>
```

Table 3. Use of OWL DL constructors in the context of the wine ontology

```
<owl:Class rdf:about="#RedWine">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Wine" />
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasColor" />
      <owl:hasValue rdf:resource="#Red" />
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

## CONCLUSION

In spite of heavy W3C support, the Semantic Web vision outlined in this article has not fully reached the status of an “inescapable” standard.

Berners-Lee’s architecture has been very criticized, in particular because it ignores some fundamental components of computer science today, from database technology (the whole world economy runs on SQL) to UML (Unified Modeling Language; UML is the standard modeling language in software engineering and, at the difference of RDF, OWL, etc., has received wide attention not only in academia but also in the professional milieu). Note, however, that some researchers are actually investigating the possibility of defining a mapping between UML and OWL-like languages. UML has, in fact, a type hierarchy comparable with OWL and a class diagrams facility that can be compared to a frame-based language; see, in this context, the comparison between UML and DAML in Baclawski et al. (2001). A general discussion about the proposals for defining transformations between UML and the Semantic Web ontology languages can be found in Falkovych, Sabou, and Stuckenschmidt (2003).

The choice of OWL as a paradigmatic language to be used for ontological work in a Web context has also raised some criticisms, and several knowledge representation specialists have challenged as hastily the endorsement of OWL by the W3C. Criticisms range from the use of a particularly cumbersome syntax, inherited from RDF/XML, to the availability of an expressive power that, from a strict knowledge representation point of view, does not seem to improve so much with respect to “traditional” frame systems like Protégé; see again the article “Ontologies and Their Practical Implementation.” We can however remark in this context that an “OWL plug-in” for Protégé has been recently implemented; see Horridge (2004) and <http://protege.stanford.edu/plugins/owl/>. It allows loading and saving OWL and RDF ontologies, editing and visualizing OWL classes and their properties, and, mainly, supporting reasoners such as the description logics classifiers.

Note that, according to OWL’s supporters, it is precisely these last characteristics that “make all the difference” between a simple frame system, which utilizes pragmatically based inference procedures, and an OWL-based reasoning tool—see, e.g., RACER (Haarslev & Möller, 2003)—that employs sound and complete inferencing algorithms supported by the description logics theory. Unfortunately, description logics have been, in turn, criticized in spite (or because) of their (too) rigorous formal framework, associated, inter alia, with a reduced expressiveness of their main reasoning component, the automatic classification mechanism.

To give only an example, nearly a printed page is needed in McGuinness et al. (2002) to demonstrate that, using the DAML+OIL definitions (DAML+OIL is the ancestor of OWL), we can infer that “Red” can be considered as a sort of “WineColor.” A plea for the use, in a Semantic Web context, of knowledge representation languages more “meaningful” than those based on a description logics approach can be found in Zarri (2002).

## REFERENCES

- Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Schneider, P. F. (Eds.). (2002). *The description logic handbook: Theory, implementation and applications*. Cambridge, UK: Cambridge University Press.
- Baclawski, K., Kokar, M. K., Kogut, P. A., Hart, L., Smith, J., Holmes, W. S., et al. (2001). Extending UML to support ontology engineering for the Semantic Web. In *Lecture notes in computer science: Vol. 2185. Proceedings of the fourth International Conference on the Unified Modeling Language, UML 2001* (pp. 342-360). Heidelberg, Germany: Springer-Verlag.
- Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., et al. (Eds.). (2004). *OWL Web Ontology Language reference* (W3C Recommendation 10 February 2004). Retrieved from <http://www.w3.org/TR/owl-ref/>
- Beckett, D. (Ed.). (2004). *RDF/XML syntax specification* (Revised; W3C Recommendation 10 February 2004). Retrieved from <http://www.w3.org/TR/rdf-syntax-grammar/>
- Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*, 284(5), 34-43.
- Biron, P. V., & Malhotra, A. (Eds.). (2001). *XML Schema part 2: Datatypes* (W3C Recommendation 02 May 2001). Retrieved from <http://www.w3.org/TR/xmlschema-2/>
- Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., & Yergeau, F. (Eds.). (2004). *Extensible Markup Language (XML) 1.0* (3rd ed.; W3C Recommendation 04 February 2004). Retrieved from <http://www.w3.org/TR/REC-xml/>
- Brickley, D., & Guha, R. V. (Eds.). (2004). *RDF Vocabulary Description Language 1.0: RDF Schema* (W3C Recommendation 10 February 2004). Retrieved from <http://www.w3.org/TR/rdf-schema/>
- Dekkers, M., & Weibel, S. (2003). State of the Dublin Core Initiative, April 2003. *D-Lib Magazine*, 9(4). Retrieved from <http://www.dlib.org/dlib/april03/weibel/04weibel.html>



Falkovych, K., Sabou, M., & Stuckenschmidt, H. (2003). UML for the Semantic Web: Transformation-based approaches. In B. Omelayenko & M. Klein (Eds.), *Knowledge transformation for the Semantic Web* (pp. 93-106). Amsterdam: IOS Press.

Haarslev, V., & Möller, R. (2003). Racer: A core inference engine for the Semantic Web. In *Proceedings of the second International Workshop on Evaluation of Ontology Tools (EON2003)*. Aachen, Germany: CEUR-WS.

Horridge, M. (2004). *A practical guide to building OWL ontologies with the Protégé-OWL plugin* (ed. 1.0, pp. 27-37). Manchester, UK: University of Manchester.

Manola, F., & Miller, E. (2004). *RDF primer* (W3C Recommendation 10 February 2004). Retrieved from <http://www.w3.org/TR/rdf-primer/>

McGuinness, D. L., Fikes, R., Hendler, J., & Stein, L. A. (2002). DAML+OIL: An ontology language for the Semantic Web. *IEEE Intelligent Systems*, 17(5), 72-80.

Smith, M. K., Welty, C., & McGuinness, D. L. (Eds.). (2004). *OWL Web Ontology Language guide* (W3C Recommendation 10 February 2004). Retrieved from <http://www.w3.org/TR/owl-guide/>

Thompson, H. S., Beech, D., Maloney, M., & Mendelsohn, N. (Eds.). (2001). *XML Schema part 1: Structures* (W3C Recommendation 02 May 2001). Retrieved from <http://www.w3.org/TR/xmlschema-1/>

Zarri, G. P. (2002). Semantic Web and knowledge representation. In *Database and expert systems applications: Proceedings of the 13th International Conference, DEXA '02* (75-79). Los Alamitos, CA: IEEE Computer Society Press.

## KEY TERMS

**DTD (Document Type Description):** A DTD is a formal description in XML declaration syntax of a particular type of document. It begins with a `<!DOCTYPE` keyword and sets out what names are to be used for the different types of markup elements, where they may occur, the elements' possible attributes, and how they all fit together. For example, a DTD may specify that every **person** markup element must have a **name** attribute, and that it can have an offspring element called **id** whose content must be text. There are many sorts of DTDs ready to be used in all kinds of areas that can be downloaded and used freely; see, e.g., MathML, for mathematical expressions; CML, Chemical Markup

Language; EDI, Electronic Data Interchange; PICS, Platform for Internet Content Selection; etc.

**OWL:** The Web Ontology Language (OWL) is a semantic markup language for publishing and sharing ontologies on the World Wide Web. OWL is developed as a vocabulary extension of RDF and is derived from the DAML+OIL Web ontology language. An OWL ontology is an RDF graph, which is in turn a set of RDF triples. OWL includes three specific sublanguages characterized by an increasing level of complexity and expressiveness: OWL Lite, OWL DL—DL stands for description logics, a particular logic-oriented knowledge representation language introduced to supply a formal foundation for frame-based systems—and OWL Full.

**RDF (Resource Description Framework):** An example of “metadata” language (metadata = data about data) used to describe generic “things” (“resources,” according to the RDF jargon) on the Web. An RDF document is a list of statements under the form of triples having the classical format: `<object, property, value>`, where the elements of the triples can be URIs (Universal Resource Identifiers), literals (mainly, free text), and variables. RDF statements are normally written into XML format (the so-called “RDF/XML syntax”).

**RDF Schema (RDFS):** Provides a mechanism for constructing specialized RDF vocabularies through the description of domain-specific properties. This is obtained mainly by describing the properties in terms of the classes of resource to which they apply: For example, we could define the **creator** property saying that it has the resource **document** as “domain” (**document** is the value or “object” of this property) and the resource **person** as “range” (this property must always be associated with a resource **person**, its “subject”). Other basic modeling primitives of RDFS allow setting up hierarchies (taxonomies), both *hierarchies of concepts*, thanks to the use of “class” and “subclass-of” statements, and *hierarchies of properties*, thanks to the use of “property” and “subproperty-of” statements. Instances of a specific class (concept) can be declared making use of the “type” statement.

**Semantic Web Architecture:** A layered architecture proposed by Berners-Lee for Semantic Web applications. In this architecture, ontologies occupy a central place: They are built on the top of the RDF (Resource Description Framework) layer, which is in turn built on the top of the XML layer (see below). The XML/RDF base constrains the particular format ontologies assume in a Semantic Web context, inheriting, e.g., all the well-known XML “verbosity.”



**XML (Extensible Markup Language):** Has been created to overcome some difficulties proper to HTML (Hypertext Markup Language), which—developed as a means for instructing the Web browsers how to display a given Web page—is a “presentation-oriented” markup tool. XML is called “extensible” because, at the difference of HTML, it is not characterized by a fixed format but lets the user design his own customized markup languages (a specific DTD, Document Type Description, see below) for

limitless different types of documents. XML is then a “content-oriented” markup tool.

**XML Schema:** A more complete way of specifying the semantics of a set of XML markup elements. XML Schema supplies a complete grammar for specifying the structure of the elements, allowing one, e.g., to define the cardinality of the offspring elements, default values, etc.

# Using Views to Query XML Documents

**Mario Cannataro**

*Magna Graecia University of Catanzaro, Italy*

**Sophie Cluet**

*INRIA Rocquencourt-Xyleme S.A., France*

**Giuseppe Tradigo**

*Magna Graecia University of Catanzaro, Italy*

**Pierangelo Veltri**

*Magna Graecia University of Catanzaro, Italy*

**Dan Vodislav**

*CNAM Cedric-Lab, Paris, France*

## INTRODUCTION

We consider using views in databases and in particular their applications to access heterogeneous and large volumes of digital documents, focusing on (1) views in databases and (2) views for data integration.

We start our paper first answering the questions what is a view and what are their applications. We give a short representation of this concept and reference Ullman (1988) and Abiteboul, Hull, and Vianu (1995) for more detailed and formal definitions in the database literature.

Views are used for several reasons to manipulate data coming from different sources through a unique and more convenient structure; e.g., to hide data (e.g., for security reasons) or to restructure data, giving a simplified *vision* of the database.

Information may be grouped in one or more collections of data depending on the database model (e.g., as a set of tables or classes in a structured model or in a graph in a semi-structured model). View facilities allow one to create an imaginary source collecting interesting data. For instance, in the case of a relational database, a view might be an imaginary table containing data coming from several tables of the database. Once views are created they can be queried by users as if they were part of the database.

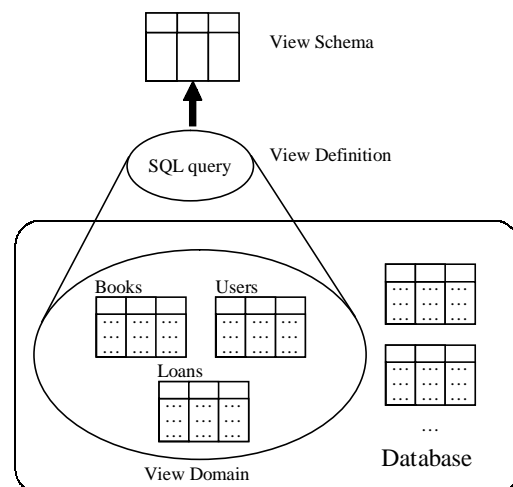
Views can be defined, queried, and modified using view definition, view query, and view manipulation languages. These three languages depend on the data model; e.g., for a relational data model, the view definition and query languages coincide with SQL, the standard query language to access to databases.

A view is created specifying a *schema*, a *domain*, and a *view definition*.

- *The view domain* is a set of sources containing information that the view represents.
- *The view schema* is the structure of the view and describes how data are presented and are queried by the users.
- *The view definition* expresses the links existing between data in the sources (actual data) and data in the view. It is expressed using the view definition language (see Figure 1).

In a simple case, the view domain is a set of data sources in the same database (e.g., a set of tables in a relational database), and the view definition language is that of the database. In more complex cases, the view

Figure 1. A view on a relational database



domain is a set of heterogeneous sources. The view schema provides a uniform access to the domain, but each source must be linked to the view schema by view definitions that depend on the source. Although there are some variations, the principle of using a view mechanism is to simplify the access to data sources. Users querying a view schema will obtain results as if they were querying one simple database. View engines, implemented in database management systems, are in charge of translating the query against the view into queries against the data.

Once defined, views can be *virtual* or *materialized*, depending on the way data are stored in the data structure associated with the view (materialized) or if the view is just a virtual representation (portion) of data sources (virtual views or, simply, views). Materializing a view consists of computing the associated query once and storing its result. Querying a virtual view implies a translation of the query using view definition to retrieve data on sources. Virtual views are used when data are updated frequently on data sources and when data cannot be replicated.

Views are materialized typically to improve query performance (Goldstein & Larson, 2001) and data availability. Querying a materialized view is much faster than querying a virtual one: A query against a materialized view is like a query against an actual database. The speed difference may be critical in applications where the query rate is high and the view is complex (e.g., with aggregate functions or joining remote sources; Chaudhuri, Krishnamurthy, Potamianos, & Shim, 1995; Srivastava, Dar, Jagadish, & Levy, 1996; Zaharioudakis, Cochrane, Lapis, Pirahesh, & Urata, 2000). Materialized views are used in data warehouses (Hammer, Garcia-Molina, Widom, Labio, & Zhuge, 1995) and distributed and multidimensional databases. Nevertheless, using materialized views implies new problems such as view selection, i.e., which views to materialize, and view maintenance, i.e., refreshing data into the view when data change (Griffin & Libkin, 1995).

Views can also be used for *data integration*. A data integration system provides a uniform query interface to a multitude of autonomous, distributed, and heterogeneous data sources, which may reside within a local database (e.g., database of an enterprise) or on distributed databases. The idea is to free the user from having to deal with several sources to have a complete answer to his queries. Queries are formulated using a uniform query language against a uniform *virtual view* of the data (also known as a *mediator*) instead of querying different data schemas. The querying system translates the query against the view into queries against the sources, retrieves results on data sources, and finally presents them to the user (Lenzerini, 2002).

Data integration systems represent a good solution to the information retrieval problem such as a search of the

Web. Users interested in finding information over the full Web use key-word-based search engines such as Altavista or Google.<sup>1</sup> Such methods are often not precise enough and return a lot of useless URLs. Answering precise queries such as “*find the names and addresses of Spanish museums that own a painting by Picasso*” requires a lot of tedious browsing through useless pages.

There have been many research projects focusing on data integration for structured and semi-structured data sources. The common idea is to provide a *global view* as an interface between end users and data sources. A query is formulated on the global view and then translated in a union of queries against the different sources. The research projects can be distinguished according to the way global view and sources are connected. In a first approach called *global-as-view* (GAV), the global view is a collection of views defined over the schema of the local sources (Cluet, Simeon, & Delobel, 1998). A query translation algorithm can be very efficient, but the global view has to be changed whenever a local source is updated. In a second approach called *local-as-view* (LAV), local sources are defined as a view over the global view. Recently, a mixed approach known as GLAV has been proposed (Lenzerini, 2002).

The emergence of the XML language (W3C, 2000) as the new (and global) standard for data exchange on the Web may change things positively by adding structure where there was none. Many research projects have proposed XML as a language to express a common interface between existing databases or to integrate in a uniform view heterogeneous data sources (Manolescu, Florescu, & Kossman, 2001). Other database-like applications, such as data monitoring, document changing notifications, active databases, and electronic commerce, have taken or will take advantage of the XML technology.

This paper presents views and database models and reports the proposals of a view system presented in Aguilera, Cluet, Milo, Veltri, and Vodislav (2002) and in Veltri (2002) as a system to query large-scale heterogeneous XML data.

## BACKGROUND

View definitions depend on the database model. In *relational databases* (Ullman, 1998), data is organized in tables (relations), each one with a schema which defines its structure (i.e., attributes and their base types such as integer, string, etc). The instance of a relation consists of a set of tuples, each containing attributes instances of the appropriate base types. For instance, the following set of relations contains data on employees and projects and belongs to the relational database of an enterprise:

## Using Views to Query XML Documents

**Employees** (SocialSecurityNb, Name, Department, Salary, Personal\_Address);  
**Projects** (ProjectId, ProjectName, Manager, ProjectSite);  
**WorkOnProject**(EmployeeId, ProjId);

Relations are queried using SQL, the standard query language for the relational model. Relational databases can be composed of a large number of tables, while users may be interested only in small portion of data. Views allow one to create a virtual relation with data relative to such attributes, loaded from the database tables, which can be accessed as a database table.

For instance, to monitor the employees that work in projects located in Paris (as well as to protect salary information from unauthorized accesses), we can create a view with the project name, the employee name, and the department name. EmployeeInParis view can be defined by means of an SQL query as:

```
CREATE VIEW EmployeeInParis AS
SELECT Name AS EmpName,
       Personal_Address AS Address,
       Department AS DeptName
FROM Employees, Projects, WorkOnProject
WHERE ProjectSite = 'Paris'
AND ProjectId = ProjId
AND EmployeeId = SocialSecurityNb
```

Users can query EmployeeInParis as if it were a base relation, using an SQL expression. For example, the following query retrieves the name of the employees in Paris:

```
SELECT EmpName
FROM EmployeeInParis
```

Processing a query against a view depends on the view implementation. As described above, if the view is materialized, then it is treated as a relation of the database and query processing is standard. If the view is virtual, it contains no data, meaning that values are not replicated, and thus represents a *virtual* source which users can query as if it had been stored in the database. For example, the previous query against EmployeeInParis is translated by the system into a query against actual data (i.e., the relations Projects, Employees, and WorkOnProject), combining the view definition with the query as follows:

```
SELECT EmpName
FROM ( SELECT Name AS EmpName,
       Personal_Address AS Address,
       Department AS DeptName
FROM Employees, Projects, WorkOnProject
WHERE ProjectSite = 'Paris'
AND ProjectId = ProjId
AND EmployeeId = SocialSecurityNb )
```

and then evaluated by the query processor as a standard query.

Views are also defined in other database models, such as the object-oriented or semi-structured ones. Views are still characterized by their schema, definition, and domain. We here sketch some examples using other models. In object-oriented databases (OODBs; Cattell, 1997), data items are *objects*, instances of classes, and are identified by a couple (*oid*, *value*), where *oid* is a unique identifier and *value* is a list of *properties* primitive (as integer, string, etc.) or complex objects (e.g., a user-defined object).

Figure 2 shows an example of a database schema in an object-oriented database: Dashed lines represent the aggregate relations between properties and classes, while full lines represent inheritance relations.

Again, views have a domain, schema, and definition. View domain is defined selecting one or more classes of the database schema. The view schema is a new class, also called a *virtual* class, and the view definition language is, in most of the proposals, an extension of OQL, the standard query language for object-oriented databases, which allows the construction of new objects. For instance, a virtual class for the employees working for a department in Paris is defined as follows:

```
virtual class EmployeeInParis from Employee
hide attributes salary
with extension EmployeeInParis =
select e from e Employee
where e.department.location = "paris"
end;
```

where the “.” notation is used to navigate through the classes relationships. Note that, thanks to inheritance, defining a view on *Employee* also gives access to objects of its subclasses, i.e., *Permanent* and *Invited* classes. Given that, it is possible to limit the access to subclasses in a view with a “hide class” command.

Figure 2. An object-oriented database schema

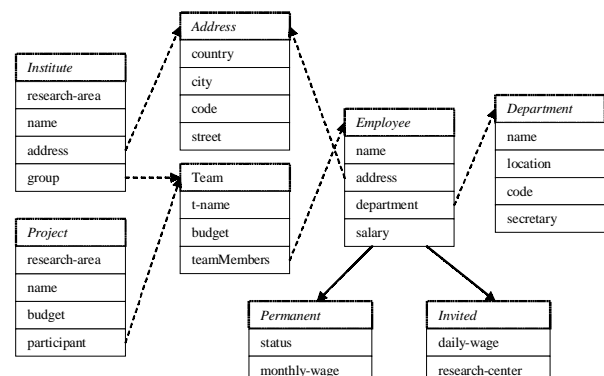
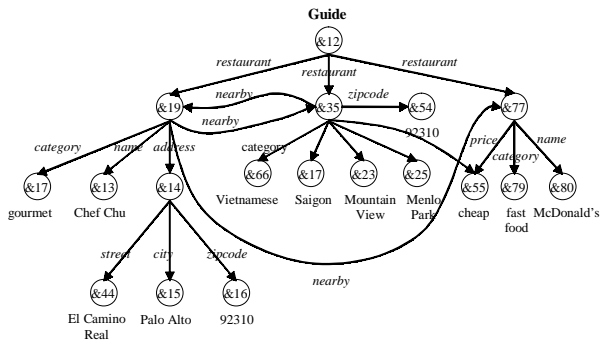


Figure 3. An OEM database



The semi-structured data model is based on the concept of data instances as a graph, with objects as the vertices and labels on the edges. Data entities are objects, each with a unique identifier *oid*. Some objects are atomic and contain a value from some base type (e.g., integer, string, gif image, audio, etc.); others are complex objects, denoted by a set of couples (*label, oid*). An example of a semi-structured database, borrowed from Abiteboul, Buneman and Suciu (2000), is shown in Figure 3 and represents restaurants in a tourist guide.

Views on semi-structured data can also be defined to increase the flexibility of a database system by adapting the data to user or application needs. In Abiteboul et al. (1995), views are defined and queried using an extension of OQL. The view is a subgraph of the graph database, and objects in the view (i.e., nodes) can be a copy of the source objects (i.e., materialized) or can refer to existing objects in the database. Again, in the latter case, queries against view are rewritten using the view definition.

XML is now the accepted standard semi-structured model (Abiteboul et al., 2000). Interestingly, it has been developed in parallel to the database efforts towards a semi-structured model. XML is used to define views to export data as XML views over the Web, keeping their internal data structure. XML views are queried using Xquery,<sup>2</sup> the standard XML query language, and then translated in the query language supported by the underlying data models. As we will see in the next section, XML views can be used as a global view for data integration.

## VIEWS ON XML DOCUMENTS

In this section we report a proposal of using views to access heterogeneous and large-scale XML documents. The proposal, presented in Aguilera et al. (2002), is implemented in a real system, Xyleme, (2001).

Let us consider the query “find the work of art of van Gogh in the Orsay museum.” Knowing the structure of the

(XML) document, the query can be formulated as expressed in Figure 4, where bounding label are target node and condition (“van Gogh” and “Orsay”) are in leaf nodes. To formulate this query users have to know the structure of the queried document. In the case of a large-scale XML database, where documents have different structures, users have to know all of them to formulate queries. Now we sketch how views can be used to avoid users having to know the structure of all queried documents.

The idea of Aguilera et al. (2002) is to create an *abstract* document (a view) that allows users to navigate through a simpler document and relay to the system the translation of the query into query on *concrete* (i.e., contained in databases) documents. Suppose documents are structured, e.g., XML documents, and contained in repositories of data. Documents can be represented as tree as in XML.

Considering a large volume of data sets and of XML documents, the view domain is a subset of data sources, while the view schema is an *abstract* tree representing a document that is not in the repository but is representative of the view domain, i.e., of a set of documents. The tree on the left side of Figure 5 is a possible abstract document representing the concrete document structure reported on the right side. The view definition, i.e., how queries on views are translated on queries against real data, of Aguilera et al. (2002) is based on a “path-to-path” mapping, i.e., a view specifies mappings between paths in the abstract and concrete tree document structures. Table 1 shows some examples of path-to-path mappings between the abstract and concrete trees of Figure 5. The “/” symbol refers to parent/child relations in the tree representing the document.<sup>3</sup>

There exist other possible mappings such as (1) label-to-label mappings, where view and documents are related by a set of mappings from an *abstract* label of a node in the view (e.g., “painting” in the abstract tree of Figure 5) and a *concrete* label of a node in some document tree (e.g., “WorkOfArt” or “painting” in the concrete document on the right side of Figure 5) and (2) building a data set of correspondence between all possible abstract subtrees and concrete trees in the data sets (Aguilera et al., 2002).

As in a standard database, the query engine uses view definition (mappings) to generate queries on documents of the database. Remember that the common goal is precision in query results.

Figure 4. Query on document structure

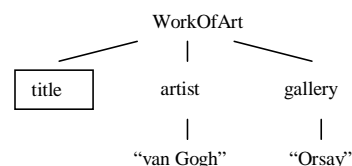




Table 1. Path-to-path mappings

Abstract path	Concrete path
painting	workOfArt artistic-Works/painting ...
painting/author	workOfArt/artist artistic-Works/painting/description/author-of-painting ...
painting/museum	workOfArt/gallery artistic-Works/exhibition artistic-Works/exhibition/gallery ...
...	...

In *path-to-path mappings* a mapping is context sensitive, e.g., node “name” belonging to path “workOfArt/period/name” has a different interpretation from the leaf node of the “person/student/name” path.

A query is formulated on the view and translated using mappings. Figure 6 shows the query of Figure 4 formulated against the view (left side) and the corresponding queries against documents obtained by a translation process using mappings of Table 1. During the translation phase, the system combines concrete paths, focusing on those generating a subtree of some concrete document in the database. The result is the union of concrete queries built from valid tree combinations. Finally, in Xyleme (2001), concrete queries are evaluated using an XQuery-like query language.

## FUTURE TRENDS

XML view mechanisms, such as that presented in this article, can be used in all applications extensively based on data exchange and data integration, where data is large

Figure 5. Abstract (view) and concrete documents

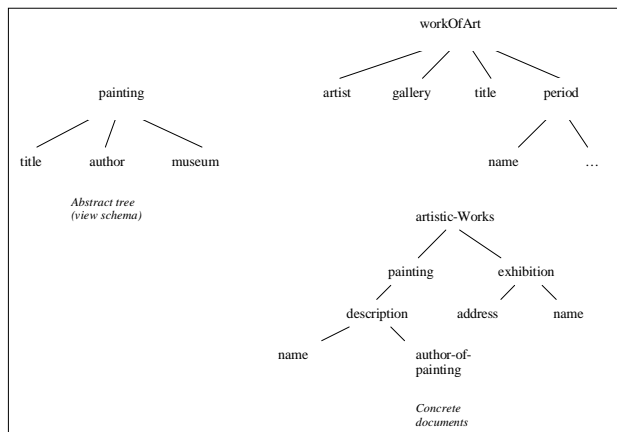
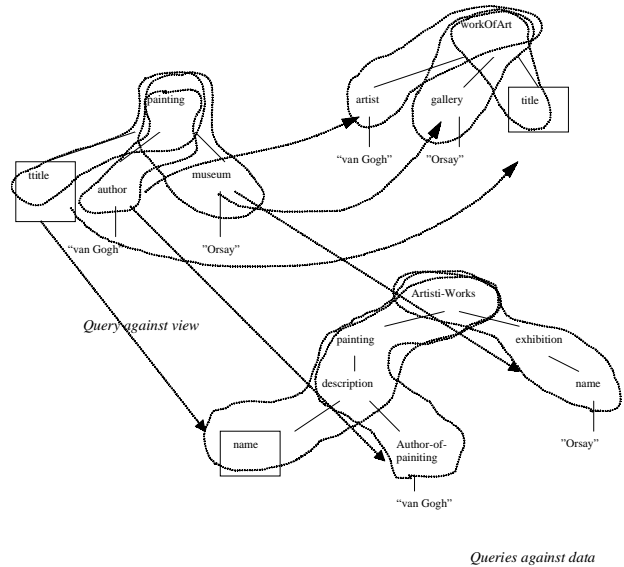


Figure 6. Query translation



and heterogeneous in contents and structure. This is what today happens, for instance, in Web applications, where *data flow* is intensive and huge in dimension.

Continuously endless data sources are producing large-scale databases, which raise a completely new set of data management problems. Examples of such continuous data flows are: phone companies’ call databases, astronomical daily digital sky surveys, audio and video surveillance devices’ data, and ISPs’ data. In this evolving scenario, new technologies have been coming out to simplify data flow management: search engines, Web-based applications, XML dialects and technologies, and grid applications. Most of these technologies rely on the XML standard, thanks to its system-independent and cross-application features. Applying the concepts presented here would allow data reorganization and optimization, allowing different views on data for different users or even application profiles, achieving data integration, and simplifying information retrieval.

Finally we imagine a world where different applications, such as business-to-business ones, may automatically exchange information using XML-based views that will automatically translate data from heterogeneous databases presenting different data schemas and structures.

## CONCLUSION

This article treats views and their use for data integration. An example of a view mechanism for structured documents is reported. The hypotheses are that document

structures are known off line and that ontology semiautomatic mechanisms are used to define views. The view mechanism reported is the one presented in Aguilera et al. (2002) and fully implemented in Xyleme (2001).

## ACKNOWLEDGMENTS

Authors are grateful to Serge Abiteboul, Tova Milo, and Sergio Greco for their comments.

## REFERENCES

- Abiteboul, S., Buneman, P., & Suciu, P. (2000). *Data on the Web*. San Francisco: Morgan Kaufmann.
- Abiteboul, S., Hull, R., & Vianu, V. (1995). *Foundations of databases*. Reading, MA: Addison-Wesley.
- Aguilera, V., Cluet, S., Milo, T., Veltri, P., & Vodislav, D. (2002). Views in a large-scale XML repository. *Very Large Databases Journal*, 11(3).
- Cattell, R. G. (1997). *The object database standard: ODMG 2.0*. San Mateo, CA: Morgan Kaufmann.
- Chaudhuri, S., Krishnamurthy, R., Potamianos, S., & Shim, K. (1995). Optimizing queries with materialized views. In *Proceedings of International Conference on Database Theory, ICDT*, Delphi, Greece (pp. 190-200). IEEE Computer Society Press.
- Cluet, S., Simeon, J., & Delobel, C. (1998). Your mediators need data conversion! In *Proceedings of the ACM SIGMOD-SIGACT-SIGART International Conference on Management of Data*. Seattle, WA: ACM Press.
- Goldstein, J., & Larson, P. (2001). Optimizing queries using materialized views. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART International Conference on Management of Data*. Santa Barbara, CA: ACM Press.
- Griffin, T., & Libkin, L. (1995). Incremental maintenance of views with duplicates. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART International Conference on Management of Data*. San Jose, CA: ACM Press.
- Hammer, J., Garcia-Molina, H., Widom, J., Labio, W., & Zhuge, Y. (1995). The Stanford data warehousing project. *IEEE Quarterly Bulletin on Data Engineering: Special Issue on Materialized Views and Data Warehousing*, 18(2).
- Lenzerini, M. (2002). Data integration: A theoretical perspective. In *Proceedings of the International Conference on Database Theory (PODS)*. Madison, WI: ACM Press.
- Manolescu, I., Florescu, D., & Kossman, D. (2001). Answering XML queries over heterogeneous data sources. In *Proceedings of 27th International Conference on Very Large Databases (VLDB)*, Rome, Italy. Morgan Kaufmann.
- Srivastava, D., Dar, S., Jagadish, H. V., & Levy, A. Y., (1996). Answering queries with aggregation using views. In *Proceedings of 22th International Conference on Very Large Databases (VLDB)*, Bombay, India. Morgan Kaufmann.
- Ullman, J. D. (1988). *Principles of database and knowledge-base systems*. New York: Computer Science Press.
- Veltri, P. (2002). *A view mechanism for large scale XML Repositories: Design and implementation*. Unpublished doctoral thesis, University of Paris XI-Orsay.
- W3C. (2000). *Extensible Markup Language (XML) 1.0 (2nd ed.) specification*. Retrieved from <http://www.w3.org/TR/REC-xml>
- Xyleme S.A. (2001). *Xyleme system*. Retrieved from <http://www.xyleme.com>
- Zaharioudakis, M., Cochrane, R., Lapis, G., Pirahesh, H., & Urata, M. (2000). Answering complex SQL queries using automatic summary table. *Proceedings of the ACM SIGMOD-SIGACT-SIGART International Conference on Management of Data*, Dallas, Texas (pp. 105-116). ACM Press.

## KEY TERMS

**Materialized View:** A materialized view physically stores data. Data is extracted from the database source (view domain) at view definition time. Data are thus duplicated in the view, and query evaluation is more efficient. Nevertheless, every database update operation must be reported on materialized views to guarantee data consistency.

**Query on Views:** A query on a view is defined using a view query language. Usually such a query language is the same as the one used for data sources. For instance, to query relational views defined on a relational database, the view query language is SQL.

**View:** A view is a logical representation of information contained in a database. It is an abstract vision of source data.

**View Definition:** Maps the view schema into view domain (data sources) using a view definition language.

## Using Views to Query XML Documents

Typically an SQL-like query language is used to map data from sources to views.

**View Domain:** Defines the data sources on which views are built and contains the origins of data.

**View Schema:** Describes how data are represented in the view both structurally and logically.

**Virtual View:** A view that does not contain data is a virtual view. Generally views are virtual and data can only be found in the data sources. Queries against such views are evaluated using view definition to retrieve results on demand from data sources (view domain).

**Xyleme:** Xyleme is a dynamic warehouse for XML data of the Web supporting query evaluation, change control, and data integration. The Xyleme Project ended in 2001 and the system is now owned by Xyleme S.A. company.

U

## ENDNOTES

- <sup>1</sup> Altavista: <http://www.altavista.com>; Google: <http://www.google.com>
- <sup>2</sup> XQuery: <http://www.w3c.org/TR/xquery>
- <sup>3</sup> See XPath: <http://www.w3c.org/TR/xpath>

# Vertical Database Design for Scalable Data Mining

**William Perrizo**

*North Dakota State University, USA*

**Qiang Ding**

*Concordia College, USA*

**Masum Serazi**

*North Dakota State University, USA*

**Taufik Abidin**

*North Dakota State University, USA*

**Baoying Wang**

*North Dakota State University, USA*

## INTRODUCTION

For several decades and especially with the preeminence of relational database systems, data is almost always formed into horizontal record structures and then processed vertically (vertical scans of files of horizontal records). This makes good sense when the requested result is a set of horizontal records. In knowledge discovery and data mining, however, researchers are typically interested in collective properties or predictions that can be expressed very briefly. Therefore, the approaches for scan-based processing of horizontal records are known to be inadequate for data mining in very large data repositories (Han & Kamber, 2001; Han, Pei, & Yin, 2000; Shafer, Agrawal, & Mehta, 1996).

On the contrary, more and more advantages of using vertical data organization have been realized. For example, it makes hardware caching work well, it makes compression easy to do, and it may greatly increase the effectiveness of the I/O device since only participating fields are retrieved instead of the whole record. The vertical decomposition of a relation also permits a number of transactions to execute concurrently. Recently, much effort has been focused on subsampling and indexing to address problems of scalability. However, subsampling requires that the subsampler knows enough about the large data set in the first place, to subsample “representatively.” That is, subsampling representatively presupposes considerable knowledge about the data. For many large data sets, such knowledge may be inadequate or nonexistent.

Index files are vertical structures and they are vertical access paths to sets of horizontal records. Some indices, such as the bit-sliced index (BSI; Chan & Ioannidis, 1998; O’Neil & Quass, 1997; Rinfret, O’Neil, & O’Neil, 2001) and encoded bitmap index (EBI; Wu, 1998; Wu & Buchmann, 1998), do address the scalability problem in many cases, but they do so at the cost of creating and maintaining additional index files separate from the data files.

Another approach, which is different from the above conceptually, is to build the whole database vertically. Such a database can be used not only for routine data management, but also for data mining. Unlike the horizontal databases, which are stored horizontally and processed vertically, vertical databases are stored vertically and processed horizontally. With other characteristics, vertical databases are shown to address the scalability issues.

## BACKGROUND

The concept of vertical data files, in fact, is not new at all. Copeland and Khoshafian (1985) presented an attribute-level decomposition storage model called DSM, similar to the attribute transposed file model (ATF; Batory, 1979) that stores each column of a relational table into a separate table. However, DSM was shown to perform well. It utilizes surrogate keys to map individual attributes together, hence requiring a surrogate key to be associated with each attribute of each record in the database. Attribute-level vertical decomposition is also

used in remotely sensed imagery, e.g., Landsat Thematic Mapper imagery, where it is called band sequential (BSQ) format. Beyond attribute-level decomposition, Wong, Liu, Olken, Rotem, and Wong (1985) presented the bit transposed file model (BTF), which further partitioned each column into bit level and utilized encoding methods to reduce the storage space. Due to the difficulty of accessing files directly in an operating system, a higher layer of accessing known as database is invented. In most cases, databases are stored horizontally, which is suitable for data retrieval but not data mining purposes. On the other hand, vertical databases can achieve both data retrieval and data mining purposes.

## MAIN THRUST OF THE ARTICLE

### Vertical Databases

In vertical databases, data are stored vertically and processed horizontally through fast, multi-operand logical operations, such as AND, OR, XOR, and complement. Predicate tree (P-tree<sup>1</sup>) is one of lossless vertical structures that can meet the requirement. P-tree is suitable to represent numerical and categorical data and has been successfully used in OLAP operations (Wang et al., 2003) and various data mining applications, including classification (Khan, Ding, & Perrizo, 2002), clustering (Denton, Ding, Perrizo, & Ding, 2002), and association rule mining (Ding, Ding, & Perrizo, 2002).

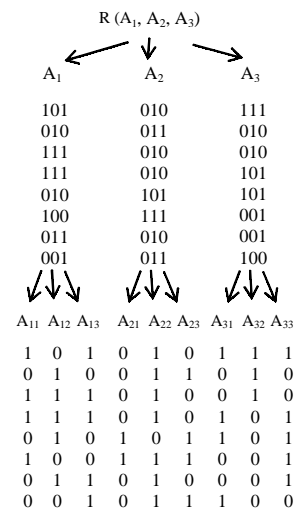
A vertical database consists of a set of P-trees rather than a set of relational tables. To convert a relational table of horizontal records to a set of vertical P-trees, the table has to be projected into columns, one for each attribute, retaining the original record order in each. Then each attribute column is further decomposed into separate bit vectors, one for each bit position of the values in that attribute. Figure 1 shows a relational table with three attributes, in which all of the attributes are numeric. Figure 2 shows the decomposition process from the relational table R to a set of bit vectors.

After the decomposition process, each bit vector is then converted into a P-tree. P-trees can be one-dimensional, two-dimensional, and multidimensional. If the data has a natural dimension, for instance, spatial data, the P-tree dimension is matched to the data dimension. Otherwise, the dimension can be chosen to optimize the compression ratio. Figure 3 shows the construction of 3 one-dimensional P-trees from the bit vectors of the second attribute A<sub>2</sub>. They are built by recording the truth of the predicate “purely 1-bits” recursively on halves of the bit vectors until purity is reached.

Figure 1. Relational table R

R (A <sub>1</sub> , A <sub>2</sub> , A <sub>3</sub> )		
5	2	7
2	3	2
7	2	2
7	2	5
2	5	5
4	7	1
3	2	1
1	3	4

Figure 2. Vertical decomposition of the table R



With various built-in engines, such as the query engine, OLAP engine, and data mining engine, vertical databases can be used to accomplish SPJ queries (Ding et al., 2002), OLAP operations (Wang et al., 2003), and various data mining applications. The detailed description of system structure is discussed in the next section.

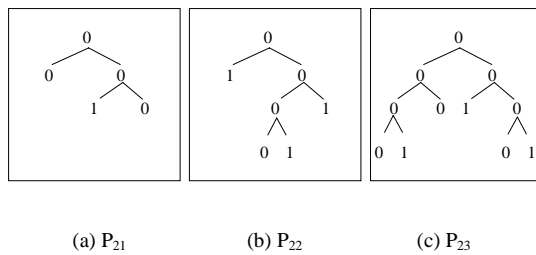
### A System Prototype

As a proof of concept, a prototype system has been developed and tested successfully for scalable data mining on the top of the vertical database concept. The multilayered software framework approach has been taken to design the prototype. The system is formally named as DataMIME™ (Serazi et al., 2004).

The layers of the system include Data Mining Interface (DMI), Data Capture and Data Integration Interface (DCI/DII), Data Mining Algorithm (DMA), and Distributed Ptree Management Interface (DPMI). DMI does counting, the most important operation for data mining provided by P-trees, including basic P-trees, value P-





Figure 3. P-trees of attributes  $A_{21}$ ,  $A_{22}$  and  $A_{23}$ 

trees, tuple P-trees, interval P-trees, and cube P-trees. DMI also provides the P-tree algebra, which has four operations—AND, OR, NOT (complement), and XOR—to implement the point-wise logical operations on P-trees for DMA. DCI/DII allows the user to capture and to integrate data to system-required format (P-tree format). The DPMI layer provides access, location, and concurrency transparency by hiding the fact that data representation may differ, resource access protocol may vary, and resources may be located in different places and shared by several competitive users. The DMA layer contains a collection of data mining tools, e.g., P-KNN (Khan et al., 2002), PINE (Perrizo et al., 2003), P-BAYESIAN (Perera, Serazi, & Perrizo, 2002), P-SVM (Pan et al., 2003), and P-ARM (Ding et al., 2002). Besides all those core layers, the system provides a graphical user interface that adds flexible user interaction with the system.

In order to comprehend how the vertical database concept affects the system, there are some key concepts that must be grasped. Unlike traditional databases, data is not stored as horizontal row-based format rather they are stored as compressed vertical P-tree format. The DPMI layer is responsible for storing and managing this P-tree-based vertical data in the system. The efficient bitwise operations on vertical data offer the scalability for data mining algorithms, and these are achieved through the DMI layer. Finally, this uniform, efficient vertical data structure at the lowest layer can take advantage of the latest hardware.

## FUTURE TRENDS

Vertical databases will become more and more important as many data sets have become extremely large. Research has shown that scanning the entire data set horizontally to be inefficient and non-scalable. Vertical databases have been exposed to be a scalable methodology that can be used to

perform fast, efficient, and effective data mining on large data sets by organizing data in vertical layouts and conducting logical operations on vertical partitioned data without scanning.

Vertical databases can be easily built in distributed systems to facilitate parallel data mining. With a large enough number of clusters, distributed vertical databases will not only lead to an efficient data mining process, but also solve the curse of high-dimension problem to a great extent.

## CONCLUSION

Horizontal data structure has been proven to be inefficient for data mining on very large sets due to the large cost of scanning. It is of importance to develop vertical data structures and algorithms to solve the scalability issue. Various structures have been proposed, among which P-tree is a very promising vertical structure. This database model is not a set of indexes but is a collection of representations of the data set itself. P-trees have shown great ability to process data containing a large number of tuples due to the fast logical AND operation without scanning (Ding et al., 2002). In general, horizontal data organization is preferable for transactional data with intended output as a relation, and vertical data structure is more appropriate for data mining on very large data sets.

## REFERENCES

- Batory, D. S. (1979). On searching transposed files. *ACM Transactions on Database Systems*, 4(4), 531-544.
- Chan, C. Y., & Ioannidis, Y. (1998). Bitmap index design and evaluation. *Proceedings of the ACM SIGMOD* (pp. 355-366).
- Copeland, G., & Khoshafian, S. (1985). Decomposition storage model. *Proceedings of the ACM SIGMOD* (pp. 268-279).
- Denton, A., Ding, Q., Perrizo, W., & Ding, Q. (2002). Efficient hierarchical clustering of large data sets using P-trees. *Proceedings of International Conference on Computer Applications in Industry and Engineering* (pp. 138-141).
- Ding, Q., Ding, Q., & Perrizo, W. (2002). Association rule mining on remotely sensed images using P-trees. *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 66-79).

- Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. *Proceedings of the ACM SIGMOD* (pp. 1-12).
- Han, J., & Kamber, M. (2001). *Data mining: Concepts and techniques*. San Francisco: Morgan Kaufmann.
- Khan, M., Ding, Q., & Perrizo, W. (2002). K-nearest neighbor classification on spatial data stream using P-trees. *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 517-528).
- O'Neil, P., & Quass, D. (1997). Improved query performance with variant indexes. *Proceedings of the ACM SIGMOD* (pp. 38-49).
- Pan, F., Wang, B., Ren, D., Hu, X., Perrizo, W. (2003). Efficient proximal support vector machine for spatial data. *Proceedings of the International Conference on Computer Applications in Industry and Engineering* (pp. 292-297).
- Perera, A., Serazi, M., & Perrizo, W. (2002). Performance improvement for Bayesian classification on spatial data with P-trees. *Proceedings of International Conference on Computer Applications in Industry and Engineering* (pp. 20-24).
- Perrizo, W., Ding, Q., Denton, A., Scott, K., Ding, Q., & Khan, M. (2003). Podium incremental neighbor evaluator for spatial data using P-trees. *ACM Symposium on Applied Computing*, (pp. 503-508).
- Rinfret, D., O'Neil, P., & O'Neil, E. (2001). Bit-sliced index arithmetic. *Proceedings of the ACM SIGMOD* (pp. 47-57).
- Serazi, M., Perera, A., Ding, Q., Malakhov, V., Rahal, I., Pan, F., et al. (2004). DataMIME™. *Proceedings of the ACM International Conference on Management of Data* (pp. 923-924).
- Shafer, J., Agrawal, R., & Mehta, M. (1996). SPRINT: A scalable parallel classifier for data mining. *Proceedings of the International Conference on Very Large Data Bases* (pp. 544-555).
- Wang, B., Pan, F., Ren, D., Cui, Y., Ding, Q., & Perrizo, W. (2003). Efficient OLAP operations for spatial data using P-trees. *Eighth ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery* (pp. 28-34).
- Wong, H. K. T., Liu, H.-F., Olken, F., Rotem, D., & Wong, L. (1985). Bit transposed files. *Proceedings of the International Conference on Very Large Data Bases* (pp. 448-457).
- Wu, M.-C. (1998). *Query optimization for selections using bitmaps* (Tech. Rep. No. DVS98-2, DVS1). Technische Universität Darmstadt, Computer Science Department.
- Wu, M.-C., & Buchmann, A. (1998). Encoded bitmap indexing for data warehouses. *Proceedings of IEEE International Conference on Data Engineering* (pp. 220-230).

## KEY TERMS

**DataMIME™:** A prototype system that has been designed and implemented on top of vertical database technology and multilayered software framework by DataSURG group at North Dakota State University, USA.

**Multilayered Software Framework:** A layer-based software environment where each layer is a group of entities dedicated to perform a particular task.

**P-Tree Algebra:** The set of logical operations, functions, and properties of P-trees. Basic logical operations include AND, OR, and complement.

**Predicate Tree (P-tree)<sup>1</sup>:** A lossless tree that is vertically structured and horizontally processed through fast multi-operand logical operations.

**Vertical Data Mining:** A process of finding pattern and knowledge from data that is organized in vertical structures, which aims to address the scalability issues.

**Vertical Database Design:** A process of developing a vertical data model, usually with intended data mining functionality that utilizes logical operations for fast data processing.

**Vertical Decomposition:** A process of partitioning a relational table of horizontal records to separate vertical data files, either to attribute level or bit level, usually retaining the original record order in each.

## ENDNOTE

- <sup>1</sup> P-tree is a patent-pending technology developed by Dr. William Perrizo's DataSURG research group at North Dakota State University.

# An XML Multi-Tier Pattern Dissemination System

**Ashraf Gaffar**

*Concordia University, Canada*

**Ahmed Seffah**

*Concordia University, Canada*

## INTRODUCTION

Patterns are widely used in several domains and are rapidly growing in numbers as an effective way of communicating knowledge between designers (Erickson, 2000). Currently, there are hundreds of HCI (human-computer interaction) patterns that are published in books and on the Internet (HCI Patterns, 2003). The sheer number of HCI patterns and the lack of a delivery system can confuse and overwhelm a novice pattern user, even when they are meant to help novice users in the first place.

## BACKGROUND

The first use of patterns is often attributed to the work of Christopher Alexander (Alexander, Ishikawa, & Silverstein, 1977). It is often claimed that in order to get the essential spirit of patterns, we should take a look at his work. The next milestone is attributed to the renowned book *Design Patterns* (Gamma, Helm, Johnson, & Vlissides, 1995), widely acknowledged in the software design community. Plenty of other patterns emerged since then, all of which are generally targeting novice designers. They aimed to provide solutions to common design problems in different contexts. This plethora of patterns started to overwhelm the users as to how to find and use this ever-growing number of patterns. So far, most efforts have focused on “generating” more patterns, but they stop short of addressing consequent phases like delivering patterns to their appropriate destinations and helping in how to effectively integrate them in a new design artifact. These essential activities are left to the user, who would consequently follow ad hoc techniques to lookup as many patterns as they can and then try to figure out how to remove redundancies, inconsistencies, and conflicts between them before attempting to put them together in a new design. This is a typical challenge of information management that has never been addressed in the pattern community before, despite the fact that elegant, well-established concepts already exist in the

database domain. Besides the efforts of adapting these concepts to the specific domain of patterns, the gap between conceptual and practical aspects needs to be bridged in the form of a software system that implements these concepts by following a structured pattern dissemination process.

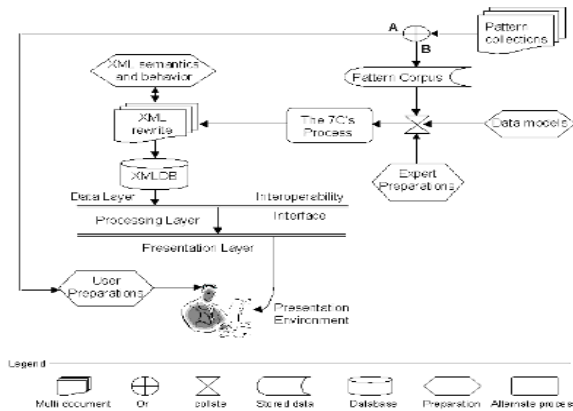
## THE SYSTEM DESIGN

Our system is based on an integrative approach of pattern dissemination that complements the scattered efforts of writing patterns. We built a delivery system in the form of a digital library based on a database of patterns, a transformation logic for any desired algorithms, pattern processing and transformation tools, as well as an interface to offer these functionalities to the user. We selected XML as the language to represent patterns for several reasons. XML is becoming the norm for data interchange and is used extensively in IT systems. Besides the XML language, several XML-based technologies are emerging to enhance XML-compatible information systems.

The system architecture is shown in Figure 1. The dataflow starts at the top right corner in the form of the existing pattern collections. Looking at the data pathways depicted in the figure, we see that the input information is now bypassed from the direct path (from pattern collections directly to the user preparations block, marked as A) into the system pre-storage phase, the pattern corpus, and to the rest of the system (marked as B).

The user preparations block, depicted as pathway A, represents the cognitive activity by the user to search for patterns and read through the text, analyze its contents, and figure out how to apply which patterns in the design. This is a fundamental observation in our study. We can evaluate the difference between the two processes, referred to as pathways A and B. If we considered them as representing the dissemination processes between pattern authors and pattern users, we can use the capability maturity model (CMM) of the Software Engineering Institute to briefly evaluate them.

Figure 1. The pattern dissemination system



- **Process A:** This process does not follow a particular approach of dissemination except for relying on users' preparations (looking up patterns, understanding them, and applying them in an ad hoc approach). We evaluated this to be a CMM level 1.
- **Process B:** As suggested by the envisioned system, there is a process in place to help users interact with patterns in a structured way. Moreover, this process relies heavily on feedback and is constantly changing, as shown in implementing and validating the 7C's process within the system (Gaffar, Sinnig, Javahery, & Seffah, 2003, Metzker, Seffah, & Gaffar, 2003). We estimate a CMM level 4.

The main modules of the system, as shown in Figure 1, are:

- The Pattern Corpus, a raw collection of patterns before being reformatted and saved in the XMLDB.
- The Data Models, used in the semantics of the XML Rewrites.
- The Expert Preparations, used as help in rewriting pattern information to ensure the integrity of the contents and avoid redundancies and inconsistencies.
- The System Process (the 7C's process), a structured method applied across the system.
- The XML Subsystem (XML Rewrites, XMLDB, and XML Semantics/Scheme), the core constituent of the system, and the backbone of the three-tiered design.
- The Processing and the Presentation layers are the middle and front tiers, respectively.

One of the main aspects of our system is to be able to automatically process the contents of patterns to allevi-

ate the user from this cognitive load. The main purpose is to have a scalable capability to effectively process a large number of patterns. Representing patterns in natural language defeats this purpose. XML is much more suitable in this regard due to its machine-readability. It is based on the fundamental concept of automatic processing. Goldfarb (Goldfarb & Prescod 2004), the inventor of SGML (the parent and superset of XML), explains that the vast majority of XML documents will be created by computer programs and processed by other computer programs, then destroyed. Humans will never see them. The first step is to rewrite patterns in an XML format. A simple XML syntax rewrite (like PLML, Pattern Language Markup Language; Fincher & Finlay, 2003) can be a small step in this direction, but—by itself—it will not do much good as will be discussed later. To achieve global interoperability, we needed to design the semantics and the behavior modeling behind the XML syntax, according to concrete data models.

The database is a core constituent of the system. An obvious choice for data store is an XMLDB. We implemented part of the system using it. The steps are as follows:

- **Phase 1, No Database System:** By temporarily including the data inside the system (hard-coding the data). We have two major prototypes: In the UPADE project, we developed a systematic approach to glue patterns together, support the integration of patterns at the high design level, and automate pattern composition. UPADE (Human-Centered Software Engineering Group, 2000) generated program elements from an extensible collection of “template” patterns.

In the second prototype (Gaffar et al., 2003; MOUDIL, 2001), we experimented with prototypes of the interface aspects, and we also hard-coded patterns inside the prototypes. At this stage we were able to test and refine the design and functionality of the system.

- **Phase 2, Flat-File Database:** To start building an independent database module, external to the system, and to connect it to the rest of the system. We used *XMLSpy* to prototype the XMLDB concept. The *XMLSpy* allowed us to build an actual, partially functional XMLDB based on flat files as the internal storage medium. The major advantage of this approach is the extreme simplicity of the system, which allowed us to further refine the fundamental concepts of the system without being carried away by the implementation details. The main drawback was its limited scalability and unacceptable performance, as will be explained in Phase 4.





- **Phase 3, Web-Based System:** To implement the functionality of the system as built around an external XMLDB on a Web-based environment. We implemented several aspects of the system using the database we have from Phase 2. Using three-tier system architecture (data, logic, and presentation tiers), DHTML, and Web technologies (CGI), we built a Web-based distributed system to test our XMLDB. We implemented the system to interact with pattern users by offering a search interface at the presentation tier.

As the user queries the system for some patterns, the system middle tier (at the Web server) processes the request by accessing the data tier at the back end and returning results by dynamically generating a Web page with search results.

## FUTURE TRENDS

XML technology is evolving in an extremely rapid pace. Just few years ago, as we started the project, the XMLDB was not supported by Oracle, an industrial norm in databases, or by Apache, an industrial norm in Web servers. The current and future trends are now determined by the new development, as explained in Phase 4 of our system development.

- **Phase 4** is to use a more advanced, fully fledged XMLDB to overcome difficulties we discovered in Phase 3. XMLSpy internally implemented the XMLDB as flat files. While invisible to the user, this solution is not scalable. It relies on the file structure of the underlying operating system and is affected by its performance. Besides, inserting into or deleting from the DB incurs the penalty of rewriting the whole file. The other major obstacle was that some XML technologies we used are still in their early specification stages and are immature for full implementation. For example, XQuery is the query language supported by XMLDB. We used it to query our XMLDB. While easy to use, it still does not support insertion or deletion, so we had to implement them manually in Phase 3.

In Phase 4, we are rebuilding the database to be less dependent on immature XML technologies while still offering all XML functionality upfront. The idea is to use newly emerging commercial or open source systems that are robust enough to support the full functionality of our XMLDB pattern system, while delegating the internal database implementation to the system, be it a native or

non-native XML database. The native XMLDB offers an XML interface to the database and stores the database internally in pure XML format. The non-native XMLDB offers an XML interface to the database, similar to the native one, but transforms it internally to a relational database or a proprietary database format before storing it physically on the hard disk.

Among the hundreds of commercially available XMLDBs today, there are three main alternatives:

1. ORACLE RDBMS with support to XMLDB. An authoritative commercial application that offers a fully functional XMLDB interface, while translating it internally to a RDBMS (a non-native XMLDB).
2. ZOPE, an open source industrial application that implements XMLDB using an internal proprietary RDBMS but provides a fully functional XMLDB interface (also a non-native XMLDB).
3. Apache-Xindice XMLDB, a powerful open source native XMLDB technology, supported by Apache (one of the best known and used Web servers).

## CONCLUSION

Patterns are useful tools that can help designers improve the quality of their design by including well-known practices of other experts. Efforts mostly focus on generating more patterns, with little or no plan on how to deliver them or how to use them in new designs. The proliferation of HCI (human-computer interaction) patterns is associated with redundancies and inconsistencies that often confuse and overwhelm the user. The database domain has long solved these problems by allowing data to be saved and reused efficiently. We have proposed an approach to complement the process of generating patterns by reducing inconsistencies and conflicts and by providing patterns in a reusable format. We have designed a process that can help in the assimilation and dissemination of patterns. We have implemented a system that—in applying the process—will help novice users receive patterns in an orderly fashion and be able to effectively use them in their design artifacts. The system relies on an underlying XML database, and the process will help feed the patterns into the database and rewrite them in a semantically reusable format. The database and the system allow us to add tools to the interface to automatically process patterns in a user-friendly design environment and to connect them to other XML- or UML-based tools.



## REFERENCES

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). *A pattern language: Towns, buildings, constructions*. New York: Oxford University Press.

Apache-Xindice Project (2003). Retrieved August 22, 2003, from <http://xml.apache.org/xindice/>

Erickson, T., (2000). Lingua Franca for design: Sacred places and pattern languages. In *DIS 2000: Proceedings of Designing Interactive Systems* (pp. 357-368). Brooklyn, NY: ACM Press.

Fincher, S., & Finlay, J., (2003). CHI 2003 Workshop report: Perspectives on HCI patterns: Concepts and tools (introducing PLML). *Interfaces*, 56, 26-28. Brooklyn, NY: ACM Press.

Gaffar, A., Sinnig, D., Javahery, H., & Seffah, A. (2003). MOUDIL: A comprehensive framework for disseminating and sharing HCI patterns (Position Paper). In *ACM CHI 2003 Workshop: Perspectives on HCI patterns: Concepts and Tools*, Ft. Lauderdale, Florida, April 2003.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns: Elements of reusable object-oriented software*. Boston: Addison-Wesley.

Goldfarb, C. F., & Prescod, P. (2004). *The XML handbook* (5th ed.). Upper Saddle River, NJ: Prentice Hall.

The HCI Patterns (2003). Retrieved on July 8, 2003, from <http://www.hcipat.terns.org/patterns.html>

Human-Centered Software Engineering Group. (2000). *UPADE (User Patterns Automated Design and Engineering)*. Retrieved May 12, 2003, from <http://hci.cs.concordia.ca/www/hcse/projects/>

Metzker, E., Seffah, A., & Gaffar, A. (2003). Towards a systematic and empirical validation of HCI knowledge captured as patterns. *Proceedings of HCI International, the 10th International Conference on Human-Computer Interaction, 1*, Crete, Greece, June.

MOUDIL Project. Retrieved on June 3, 2003, from <http://hci.cs.concordia.ca/moudil/>

## KEY TERMS

**Capability Maturity Model (CMM):** A model defined by the Software Engineering Institute and used to assess the capability and the maturity of a software process. The

CMM levels range from 1 (initial, ad hoc) to 5 (optimizing, process improvement).

**CGI:** Common gateway interface, a standard protocol used on the World Wide Web, which allows Web pages and distributed applications to communicate with a Web server and request some services on the server side.

**DHTML:** Dynamic Hypertext Markup Language, the new standard of HTML, which adds scripting language and styling capabilities to HTML, hence allowing users more interactive sessions and layout changes of the Web pages.

**HCI Patterns:** Human-computer interaction patterns, which are patterns that focus on how users interact with computers via the interface of the application.

**Interoperability Interface:** An interface between XML documents and other software programs. The interoperability stems from the fact that the XML documents can be read and operated upon by any software application designed to use this interface.

**Pattern Processing and Transformation Tools:** These are software tools that can access patterns written in XML format and apply arbitrary transformation logic on them to produce desired results.

**Presentation Environment:** An integrated development environment (IDE) that displays selected patterns to the user when retrieved from the database and offers different pattern processing and transformation tools.

**Transformation Logic:** XML-based documents can be read by other applications. Their contents can be manipulated according to desired rules and algorithms and then transformed to new documents with partial or different contents. These rules are referred to as transformation logic and are determined by the applications.

**Web-Based Environment:** A new paradigm in which applications are meant to be distributed over the World Wide Web, offering universal interfaces and remote accessibility and processing across the Internet.

**XML Rewrites:** Documents that are not originally written in XML can be rewritten in XML format by adding the necessary syntax and semantics to them.

**XMLDB:** A database that accepts XML documents for storage, and allows for retrieving them in their original XML format. XMLDB can physically store XML data in their original format or can transform them internally into relational tables, which are then reverted to their XML format upon retrieval.

**XQuery:** An XML query language, the specification of which is defined by the W3C, the authorities body behind XML. It can access XML databases and documents and offer different querying capabilities.