SIGMOD Officers, Committees, and Awardees

Chair

Raghu Ramakrishnan
Yahoo! Research
2821 Mission College
Santa Clara, CA 95054
USA
<First8CharsOfLastName AT yahoo-inc.com>

Vice-Chair

Yannis Ioannidis
University of Athens
Department of Informatics & Telecom
Panepistimioupolis, Informatics Buildings
157 84 Ilissia, Athens
HELLAS
<yannis AT di.uoa.gr>

Secretary/Treasurer

Mary Fernández
ATT Labs - Research
180 Park Ave., Bldg 103, E277
Florham Park, NJ 07932-0971
USA
<mff AT research.att.com>

SIGMOD Officers, Committees, and Awardees (continued)

SIGMOD Edgar F. Codd Innovations Award

For innovative and highly significant contributions of enduring value to the development, understanding, or use of database systems and databases. Until 2003, this award was known as the "SIGMOD Innovations Award." In 2004, SIGMOD, with the unanimous approval of ACM Council, decided to rename the award to honor Dr. E.F. (Ted) Codd (1923 - 2003) who invented the relational data model and was responsible for the significant development of the database field as a scientific discipline. Recipients of the award are the following:

Michael Stonebraker (1992)	Jim Gray (1993)	Philip Bernstein (1994)
David DeWitt (1995)	C. Mohan (1996)	David Maier (1997)
Serge Abiteboul (1998)	Hector Garcia-Molina (1999)	Rakesh Agrawal (2000)
Rudolf Bayer (2001)	Patricia Selinger (2002)	Don Chamberlin (2003)
Ronald Fagin (2004)	Michael Carey (2005)	Jeffrey D. Ullman (2006)
Jennifer Widom (2007)	Moshe Y. Vardi (2008)	

SIGMOD Contributions Award

For significant contributions to the field of database systems through research funding, education, and professional services. Recipients of the award are the following:

Maria Zemankova (1992)	Gio Wiederhold (1995)	Yahiko Kambayashi (1995)
Jeffrey Ullman (1996)	Avi Silberschatz (1997)	Won Kim (1998)
Raghu Ramakrishnan (1999)	Michael Carey (2000)	Laura Haas (2000)
Daniel Rosenkrantz (2001)	Richard Snodgrass (2002)	Michael Ley (2003)
Surajit Chaudhuri (2004)	Hongjun Lu (2005)	Tamer Özsu (2006)
Hans-Jörg Schek (2007)	Klaus R. Dittrich (2008)	

SIGMOD Jim Gray Doctoral Dissertation Award

SIGMOD has established the annual SIGMOD Jim Gray Doctoral Dissertation Award to *recognize excellent research by doctoral candidates in the database field.* This award, which was previously known as the SIGMOD Doctoral Dissertation Award, was renamed in 2008 with the unanimous approval of ACM Council in honor of Dr. Jim Gray. Recipients of the award are the following:

- **2008** *Winner*: Ariel Fuxman (advisor: Renee J. Miller), University of Toronto.
  Honorable Mentions: Cong Yu (advisor: H. V. Jagadish), University of Michigan;
  Nilesh Dalvi (advisor: Dan Suciu), University of Washington.
- **2006** *Winner*: Gerome Miklau, University of Washington.
  Runners-up: Marcelo Arenas, Univ. of Toronto; Yanlei Diao, Univ. of California at Berkeley.
- **2007** *Winner*: Boon Thau Loo, University of California at Berkeley.
  Honorable Mentions: Xifeng Yan, UIUC; Martin Theobald, Saarland University.

A complete listing of all SIGMOD Awards is available at: **http://www.sigmod.org/awards/**

[Last updated on May 31, 2009]

Editor's Notes

Welcome to the March 2009 issue of SIGMOD Record. We begin the issue with a (goodbye) message from Raghu Ramakrishnan, SIGMOD Chairperson.

We continue the issue with information about the process for the SIGMOD Elections, by Tamer Ozsu, chair of the nominating committee, followed by the statements of the candidates:

- Amr El Abbadi and Yannis Ioannidis (candidates for SIGMOD Chairperson).
- Anastasia Ailamaki and Christian S. Jensen (candidates for SIGMOD Vice-Chair).
- Alexandros Labrinidis and Wang-Chiew Tan (candidates for SIGMOD Secretary/Treasurer).

The first article of this issue is a reprint of "Derivability, Redundancy and Consistency of Relations Stored in Large Data Banks", by E. F. Codd, which was published in August 19, 1969 as IBM Research Technical Report RJ599. This article is considered to be the *first paper on the relational model*. With it, we inaugurate a new column, that will celebrate the **40-year anniversary of the relational model** and will run throughout 2009. I would like to take a moment to thank IBM Research for graciously (and very expeditiously) providing permission to reprint this article.

Next, we have two regular **research articles**. The first one, by Laender, Moro, Nascimento, and Martins provides a nice characterization study of XML Schemas available on the Web. The second one, by Florescu and Kossmann argues in favor of using real cost (i.e., in $) as an optimization metric when considering database performance.

The **Database Principles Column** (edited by Leonid Libkin) features one survey article, by Barcelo, on relational data exchange, i.e., on transforming data from a source to a target schema.

We continue with an article in the **Surveys Column** (edited by Cesar Galindo-Legaria), on "Data Integration for Mashups" by Lorenzo, Hacid, Benatallah, and Paik. The article analyzes the advantages and disadvantages of the various mashup tools with respect to the data integration aspect.

The **Distinguished Profiles in Data Management Column** (edited by Marianne Winslett) features an interview of Gerhard Weikum, who is a Research Director at the Max-Planck Institute for Informatics in Saarbrucken, Germany. Gerhard is an ACM Fellow and the president of the VLDB Endowment. Read Gerhard's interview to find out (among many other things) why we should go for the grand challenges, why SQL is too powerful, and how to have a big research group in Germany.

Next is the **Open Forum Column**, which is meant to provide a forum for members of the broader data management community to present (meta-)ideas about non-technical issues and challenges of interest to the entire community. In this issue, we publish a Call for Opinions, on *the role of workshops in the dissemination of research results*.

Next we have an important **Announcement** from the Computing Research Association (CRA) about the newly established *Computing Innovation Fellows Project*, which provides a very good opportunity for new PhDs in computing-related disciplines.

(continued on next page)

We close the issue with multiple **Calls for Participation**:

- The 2009 SIGMOD/PODS Conference, held in Providence, RI, June 28th till July 2nd.
- SIGMOD 2009 Workshops:
  - KEYS 2009: The first International Workshop on Keyword Search for Structured Data.
  - MobiDE 2009: The eighth International ACM Workshop on Data Engineering for Wireless and Mobile Access (10-Year Anniversary).
  - WebDB 2009: The 12th International Workshop on the Web and Databases.
- ER 2009: The 28th International Conference on Conceptual Modeling.

In the past, whenever a new issue of SIGMOD Record became available, a message was posted on `sigmod.org` and on the SIGMOD RSS feed. Given the push for online membership, we will be implementing additional **notification mechanisms** when a new issue of SIGMOD Record becomes available on `sigmod.org/record`:

- an email message with go out to all SIGMOD members, and
- a "tweet" will be posted on `http://twitter.com/sigmodrecord`, so if you have a twitter account, simply follow *sigmodrecord* to get updates.

In this way, you will be able to get an issue as soon as it becomes available, even if you are not signed up for the print version.

Alexandros Labrinidis
May 2009

Chair's Message

It has been a satisfying and challenging four years, serving on the SIGMOD EC, and I could not have asked for better colleagues in arms than Mary Fernandez as Treasurer and Yannis Ioannidis as Vice-Chair, not to mention the many, many people on the extended EC who collectively run all aspects of ACM SIGMOD, from the conferences to DISC, to the SIGMOD Record.

As I look back, there are some accomplishments I'm proud of. We've had a vibrant series of SIGMOD/PODS conferences; we've a strong competition for the best doctoral dissertation in database management; we've sound, self-renewing processes in place to identify members of the community for recognition through awards, and to select representative papers to nominate to CACM; we've explored innovative reviewing and publishing models such as rollover of papers and experimental repeatability; we've revamped our membership levels and continue to provide unique member services such as DISC and support for undergraduate research and travel to conferences; and we've a new travelling researcher program in place. Most importantly in these difficult economic times, we're in sound financial shape, with the funds to continue to carry forth our charter to support and disseminate database research and education.

Many challenges remain. We continue to struggle with the increase in the number of papers being submitted and the corresponding increase and geographical spread of our conference reviewing committees. We need to understand the right balance between a culture of conference publication and archival journal publication, and engage with our colleagues at VLDB on programs like the conference-journal hybrid they are developing, as and when this is appropriate. We need to find ways to use our reach to improve member services further, and in particular, to provide improved mentoring for the younger people in the field.

It is time, however, for the old order to change, and yield place to new. In other words, it's election time. I'm grateful to Tamer Ozsu for leading the effort to assemble the slate for the elections. I'm excited to see just how strong it is—SIGMOD will be in very good hands indeed! Just which hands, however, is up to you, the members. Please take a moment to consider the backgrounds and statements of the candidates, and cast your ballot. And thank you—it was a privilege to serve as Chair of SIGMOD and to see all the work and the many people who make it such a great organization.

Sincerely,
Raghu Ramakrishnan

SIGMOD Officers Election Process

As I am sure you are well-aware, we are in the midst of an election to identify the next set of officers to run SIGMOD. It is even possible that you may have learned the results of the elections by the time you read this. In this short note, I explain the process that we followed in this process.

In Fall 2008, the current SIGMOD Chair, Raghu Ramakrishnan, asked me to chair the Nominating Committee. Together we selected the other members. In addition to the two of us, the Committee consisted of Susan Davidson (University of Pennsylvania), Gustavo Alonso (ETH Zürich), and Kyu-Young Whang (KAIST).

The Committee issued a public call for nominations through DBWORLD. We did get some nominations in response to our call, and committee members also gathered lists of potential candidates. We conducted a number of rounds of email discussions before we developed a short list. In the end, I was charged with contacting colleagues on the short list and asking them if they would be willing to be candidates. ACM regulations require at least two candidates for each position, and at the end of this process the election slate consisted of the following colleagues:

Chair candidates:
Amr El Abbadi, University of California, Santa Barbara
Yannis Ioannidis, University of Athens

Vice-Chair candidates:
Anastassia Ailamaki, École Polytechnique Fédérale de Lausanne
Christian Jensen, Aalborg University

Secretary/Treasurer candidates:
Alexandros Labrinidis, University of Pittsburgh
Wang-Chiew Tan, University of California, Santa Cruz

The Nominating Committee is very pleased with the slate that we have been able to present to the SIGMOD membership, and we feel confident that SIGMOD will be in very good hands for the next four years regardless of the results of the election. The candidates are active members of our community and represent the growing international nature of SIGMOD.

M. Tamer Özsu
May, 2009

CANDIDATE FOR ACM SIGMOD CHAIRPERSON

PROF. AMR EL ABBADI

UNIV. OF CALIFORNIA, SANTA BARBARA, USA

http://www.cs.ucsb.edu/~amr

Professional Experience:

- Assistant, Associate and Professor, Univ. of California, Santa Barbara, 1987-present
- Chair of Computer Science Department, Univ of California, Santa Barbara, 2007--present
- Director, Univ of Calif, Education Abroad Program, American University in Cairo, Egypt, 2002 – 2003.
- PhD, Department of Computer Science, Cornell University, 1987.

Current areas of professional interest:

Data Management, Distributed Information Systems, Data Stream, Digital Libraries, Data Privacy, Fault-tolerance, Hardware support for Databases.

ACM activities:

- ACM Distinguished Lecturer, Distinguished Speakers Program, 1991 – 1992.
- Group Leader, SIGMOD Conference, 2005.
- Program Committee Member, ICDE, ICDT, EDBT, PODS, SIGMOD, ACM GIS etc., Many years.

Membership and offices in related organizations:

- Board of Trustees, VLDB Endowment, 2002-2008.
- Associate Editor, Information Systems, 1994 – 2001
- Editor for Information Processing letters 1998-2002.
- Americas Program Chair, Very Large Data Bases Conference, 2001
- Vice Chair for ICDCS 1999, Vice Chair ICDE 2002.

Awards received:

- IEEE ICDE Best Paper Award, 2002
- UCSB Senate Outstanding Mentorship Award, 2007.

Statement:

Data intensive computing and data centric thinking are central to many of the current and future innovations in computer science, and SIGMOD is critical to this development. As our community expands, and as the significance and impact of managing, understanding and analyzing data increases, it is important for the SIGMOD community to embrace and collaborate with diverse international and research communities. Our goal is not only to maintain our leading research edge, but also to ensure that the processes with which our main products, namely research, are chosen and published, adapt and evolve. These are very exciting times with many challenges, and the community has already started to explore innovative ways to ensure that this process is scalable, global and successful. As an educator, I have been a strong believer that the training of researchers is crucial for success in both academia and industry. If elected; I will explore novel ways to more fully incorporate young talents into the SIGMOD community, thus benefiting from their expertise and enthusiasm and assist in their own development and maturity. As an advocate of broad research interactions, I have a long history of serving on ACM and

IEEE program committees. If elected, I would bring diverse experiences to this position, be they as a researcher, an educator, a mentor, or as a citizen of the world with diverse experiences in different countries. Serving on the VLDB Endowment provided me with many insights into the challenges and demands of our community and to observe first hand that synergy among our different publication venues and communities is crucial for addressing many of the challenges we face, especially as we scale in our interests, numbers and research output. If elected, I will be honored to be part of this process, and to ensure its success.

CANDIDATE FOR ACM SIGMOD CHAIRPERSON

PROF. YANNIS IOANNIDIS

UNIV. OF ATHENS, HELLAS (GREECE)

http://www.di.uoa.gr/~yannis

Professional Experience:

- Professor, Univ. of Athens, Hellas (Greece), 2001-present
- Associate Professor, Univ. of Athens, Hellas (Greece), 1997-2001
- Professor, Univ. of Wisconsin-Madison, Madison, WI, 1998-1999 (on leave)
- Associate Professor, Univ. of Wisconsin-Madison, Madison, WI, 1993-1998 (last year on leave)
- Assistant Professor, Univ. of Wisconsin-Madison, Madison, WI, 1986-1993

Current areas of professional interest:

- Query processing and optimization in distributed architectures
- Personalization and contextualization of data/info management
- Digital libraries, cultural and scientific information systems
- Information retrieval, databases and the web
- User interfaces

ACM activities:

- Vice-Chair, ACM SIGMOD, 2005-present.
- Associate Editor, ACM SIGMOD Digital Symposium Collection (DiSC), 1998-2006
- Workshops Chair, ACM SIGMOD Conference, 2006-2009
- Program Committee Member, ACM SIGMOD, ACM CHI ("notes" associate chair), ACM PODS, many years

Membership and offices in related organizations:

- Member of Board of Trustees, VLDB Endowment, 1998-2003
- Member of Executive Committee (Sigmod Liaison), IEEE Technical Committee on Data Engineering, 2005-present
- Member, EDBT Endowment, 2006-present
- Program Committee (co-)Chair, ICDE'09, ADBIS'07, EDBT'06, VLDB'02, VDB'98, SSDBM'97
- Associate Editor, Information Systems (1993-present), VLDB Journal (1997-2005), Int'l Journal on Digital Libraries (2002-present), Journal of Intelligent Information Systems (1990-present), ...

Awards received:

- ACM Fellow, 2004
- "Xanthopoulos-Pneumatikos" Award for "Outstanding Academic Teaching", 2006
- VLDB "10-Year Best Paper Award", 2003
- Univ. of Wisconsin "Chancellor's Distinguished Teaching Award", 1996
- NSF "Presidential Young Investigator Award", 1991

Statement:

SIGMOD has had a long sequence of inspiring and dedicated chairs that have brought it in its current position of scientific leadership and financial stability. If honored to be elected as their successor, I will make every effort to continue their tradition and serve the community, pushing SIGMOD in new directions while maintaining its strengths. There are several new challenges and many stimulating opportunities in front of SIGMOD, and I hope that my experience as vice-chair the past four years will help me deal with both most effectively.

In the midst of a global financial crisis, finding the right balance between supporting important initiatives and remaining financially safe is a priority. Increasing membership and identifying new sponsorship schemes are two income-raising strategies I intend to work on. Research-wise, data management is now in the critical path of most scientific and societal activities; in conjunction with rapid advances in other technologies, this unprecedented breadth of data-centric applications gives rise to several new research problems that are difficult to solve, and brings our field to an exciting turning point. I will work towards putting SIGMOD in the driver seat of all relevant developments, seeking to join forces and form strategic alliances with other SIGs and other professional societies when beneficial. Finally, I will continue several efforts initiated during my previous term, such as the Traveling Speakers Program and making all/most DiSC material available on-line for members, and will start discussions on new important issues.

CANDIDATE FOR ACM SIGMOD VICE CHAIRPERSON

PROF. ANASTASIA AILAMAKI

ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE, SWITZERLAND
CARNEGIE MELLON UNIVERSITY, USA

http://people.epfl.ch/anastasia.ailamaki
http://dias.epfl.ch

Professional Experience:

- Assistant and Associate Professor, Carnegie Mellon University, Pittsburgh, PA, 2001-2007
- Adjunct Associate Professor, Carnegie Mellon University, Pittsburgh, PA, 2008-present
- Professor, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 2008- present

Current areas of professional interest:

Database Management Systems, Computer Architecture, Scientific Data Management, Computer Systems.

ACM activities:

- Associate Editor, ACM SIGMOD Digital Symposium Collection (DiSC), 2002-2003.
- Electronic Proceedings Chair, ACM SIGMOD Int'l Conf. on Management of Data, 2002.
- Program Committee Chair, Demo track, ACM SIGMOD Int'l Conf. on Management of Data, 2006.
- Program Committee Chair, General Chair, and Organizer, ACM SIGMOD Workshop on Data Management on Modern Hardware (DaMoN). 2005-now
- Program Committee Member, SIGMOD, SIGPLAN/SIGARCH/SIGOPS ASPLOS, SIGPLAN TRANSACT, etc. 2000-present.

Membership and offices in related organizations:

- Publications Chair (2006-07) and Chair (2008-present), IEEE Workgroup on Self-Managing Database Systems.
- Associate Editor, IEEE Data Engineering Bulletin, 2006-2008.
- Vice PC Chair, IEEE International Conference on Data Engineering, 2004 and 2007
- Program Committee Member, ICDE, VLDB, SBBD, etc. 2001-present

Awards received:

- OTHER: European Science Foundation: European Young Investigator Award, 2007
- OTHER: Alfred P. Sloan Foundation: Sloan Research Fellow, 2005
- OTHER: Six Best Paper Awards (VLDB 2001, ICDE 2004, FAST 2005, etc.), 2009
- OTHER: Anthony C. Klug NCR Graduate Fellow, 1998

Statement:

Nowadays data management is more ubiquitous than ever, and has quickly become the key technology behind every challenging application. New research problems arise, ranging from requirements of diverse scientific applications, to new exciting problems arising from emerging businesses, to the revolutionary hardware and storage components that constitute our platforms.

I am deeply honored to be nominated as an ACM SIGMOD officer in these fascinating times. As the leading community for storing, retrieving, and understanding data, SIGMOD should be a strong pole of attraction for other disciplines. If elected, I will try to increase SIGMOD membership counts by promoting interdisciplinarity, thereby enhancing diversity while contributing to financial prosperity. I plan to entice knowledgeable scientists who daily research and resolve challenging data management problems from a non-computer-science standpoint, as their work is tied to their particular field. I would also like to contribute to ongoing efforts towards improving quality of reviewing submitted work to SIGMOD. Finally, I plan to start a systematic effort to facilitate student attendance to conferences, as well as invent relatively informal but highly interactive ways for young researchers, new faculty, and underrepresented groups (e.g., women and scientists from remote geographic areas) to showcase their work.

CANDIDATE FOR ACM SIGMOD VICE-CHAIR

PROF. CHRISTIAN S. JENSEN

AALBORG UNIVERSITY, DENMARK

http://www.cs.aau.edu/~csj

Professional Experience:

- Professor, Aalborg University, DK, 1998-present
- Associate Professor, Aalborg University, DK, 1994-1997
- Assistant Professor, Aalborg University, DK, 1990-1994
- Visiting Scientist, Google Inc, Mountain View, CA, USA, 2008-2009
- Visiting Professorial Fellow, NUS, Singapore, 2002
- Visiting Professor, Associate Professor, Scholar, University of Arizona, 1999, 1994-1995, 1992, 1991

Current areas of professional interest:

Temporal, spatial, and spatio-temporal data management; mobile and ubiquitous data management; mobile services; data warehousing; query processing and indexing

ACM activities:

- Associate Editor, ACM Transactions on Database Systems, SIGMOD, 2001-2007
- European Coordinator and Chair, Special Sessions, Panels, Tutorials, ACM SIGMOD Conf., 1997-2006
- PC member, ACM SIGMOD Conference (multiple times), 1994-2007

Membership and offices in related organizations:

- Editor-in-Chief w. P. Bernstein and K.-Y. Whang, The VLDB Journal, 2008-present
- Member, Board of Trustees of the VLDB Endowment, 2004-2009
- Associate Editor, IEEE Transactions on Knowledge and Data Engineering, 1997-2001
- PC Chair and Co-Chair, DMSN 2008, TIME 2008, MDM 2007, MobiDE 2006, VLDB 2005, EDBT 2002, SSTD 2001; PC Vice-Chair, ICDE 2008, 1998

Awards received:

- IEEE Fellow
- Telenor's Nordic Research Award
- Ib Henriksen's Research Award
- Honorary Professor, Cardiff University, UK

Statement:

SIGMOD is an outstanding organization and it is a privilege to be given the opportunity to run for Vice Chair. If elected, my main objectives will be to understand and meet the needs of the community as best as I am able. I am committed to continuing to innovate SIGMOD.

The SIGMOD Conference is of central importance to the SIGMOD community. Recent years have seen increased dissatisfaction with the review process, and many attempts have been made at reengineering the review process within the constraints of a conference setting. However, if the quality of the reviewing itself is not high, such reengineering is ineffective. Many have observed that there are few rewards for good reviewing. I propose that SIGMOD initiates an effort to find ways of rewarding quality reviewing, to be introduced gradually and evaluated in a systematic manner.

I will work to integrate social networking tools into the SIGMOD web site in a way consistent with the high quality of SIGMOD, to ensure that the site reflects the breadth of our community and becomes more dynamic.

In particular, I believe that the very advances brought about by members of our community in the area of web data offer opportunities for further improving the SIGMOD web site. As one example, we should aim for a much more dynamic site that members of the community will want to visit frequently. To achieve that, additional and relevant content, including member-generated content, should be enabled. We should also aim for new ways of establishing an increased SIGMOD presence on the web.

Next, SIGMOD can and should do more to involve and provide services to its members and to the database community across the globe. For example, regional columns may be introduced in SIGMOD Record and on the SIGOMD web site.

Throughout my twenty years as a database researcher, I have had substantial collaborations across Asia, Europe, and the US; and I have spent substantial time in each. As a result, I am aware of the many different perspectives within our profession, and I will work hard to represent and honor them all. I have built a sizable research group in my department and I have served in leadership roles for top conferences as well as on some 140 PCs. That is the kind of commitment I will bring to SIGMOD.

CANDIDATE FOR ACM SIGMOD SECRETARY/TREASURER

PROF. ALEXANDROS LABRINIDIS

UNIV. OF PITTSBURGH, USA

http://www.cs.pitt.edu/~labrinid

Professional Experience:

• Associate Professor, University of Pittsburgh,	Pittsburgh, PA, USA	2008 –
• Assistant Professor, University of Pittsburgh,	Pittsburgh, PA, USA	2002 – 2008
• Adjunct Associate Professor, Carnegie Mellon University,	Pittsburgh, PA, USA	2008 –
• Adjunct Assistant Professor, Carnegie Mellon University,	Pittsburgh, PA, USA	2002 – 2008

Current areas of professional interest:

Web Data Management; Data Stream Management Systems; User-Centric Data Management

ACM activities:

- Editor, SIGMOD Record, 2007 –
- Information Director, SIGMOD, 2004 – 2006
- Associate Editor, SIGMOD Record, 2002 – 2006
- Webmaster, SIGMOD Record, 1997 – 2002

Membership and offices in related organizations:

- Program Committee Member: ICDE (2004 – 2010), VLDB (2004, 2007, 2008), EDBT (2004, 2008) and many other conferences & workshops
- Program Committee Co-Chair: WebDB 2009, GSN 2006, DMSN 2005, MobiDE 2005, DMSN 2004
- General Co-Chair/Vice-Chair: MobiSensors 2007, DMSN 2006, MobiDE 2006

Awards received:

- OTHER: NSF Career Award (2008)
- OTHER: Innovation in Education Award – University of Pittsburgh (2007)

Statement:

I am honored to be nominated and thrilled at the opportunity to serve as secretary/treasurer of SIGMOD. I have been involved as a volunteer for SIGMOD in various positions for the past 12 years; currently, I am the Editor of SIGMOD Record (if elected, I will step down as Editor to focus on my new duties).

Despite the tough times the global economy is in, SIGMOD's finances are in a very healthy state, thanks to the hard work and prudence of the past elected officers and conference organizers. As such, although it will not be business as usual in the coming years, I am confident that we will maintain all member benefits and start important new initiatives without increasing membership fees.

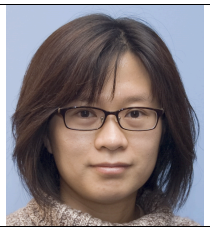
If elected, I will work for our community to achieve increases on three fronts: use of the Web, our membership, and our visibility. First, we can revitalize sigmod.org, by including user-generated content. Secondly, we should continue and expand our efforts to increase SIGMOD membership in the US and internationally (one target demographic: undergraduate students). Finally, we should increase the visibility of the data management community to the sciences in general and to computer science in particular.

CANDIDATE FOR ACM SIGMOD SECRETARY/TREASURER

PROF. WANG-CHIEW TAN

UNIV. OF CALIFORNIA AT SANTA CRUZ, USA

http://www.cs.ucsc.edu/~wctan



Professional Experience:

- Associate Professor, Univ. of California at Santa Cruz, Santa Cruz, CA, 2008 - present
- Assistant Professor, Univ. of California at Santa Cruz, Santa Cruz, CA, 2002-2008
- Visiting Scientist, IBM Almaden Research Center, Sep 2008 - Mar 2007, Sep - Dec 2008
- Visiting Scientist, Institute for Infocomm Research, Singapore, July 2004

Current areas of professional interest:

Data management with emphasis on Data Provenance and Information Integration.

ACM activities:

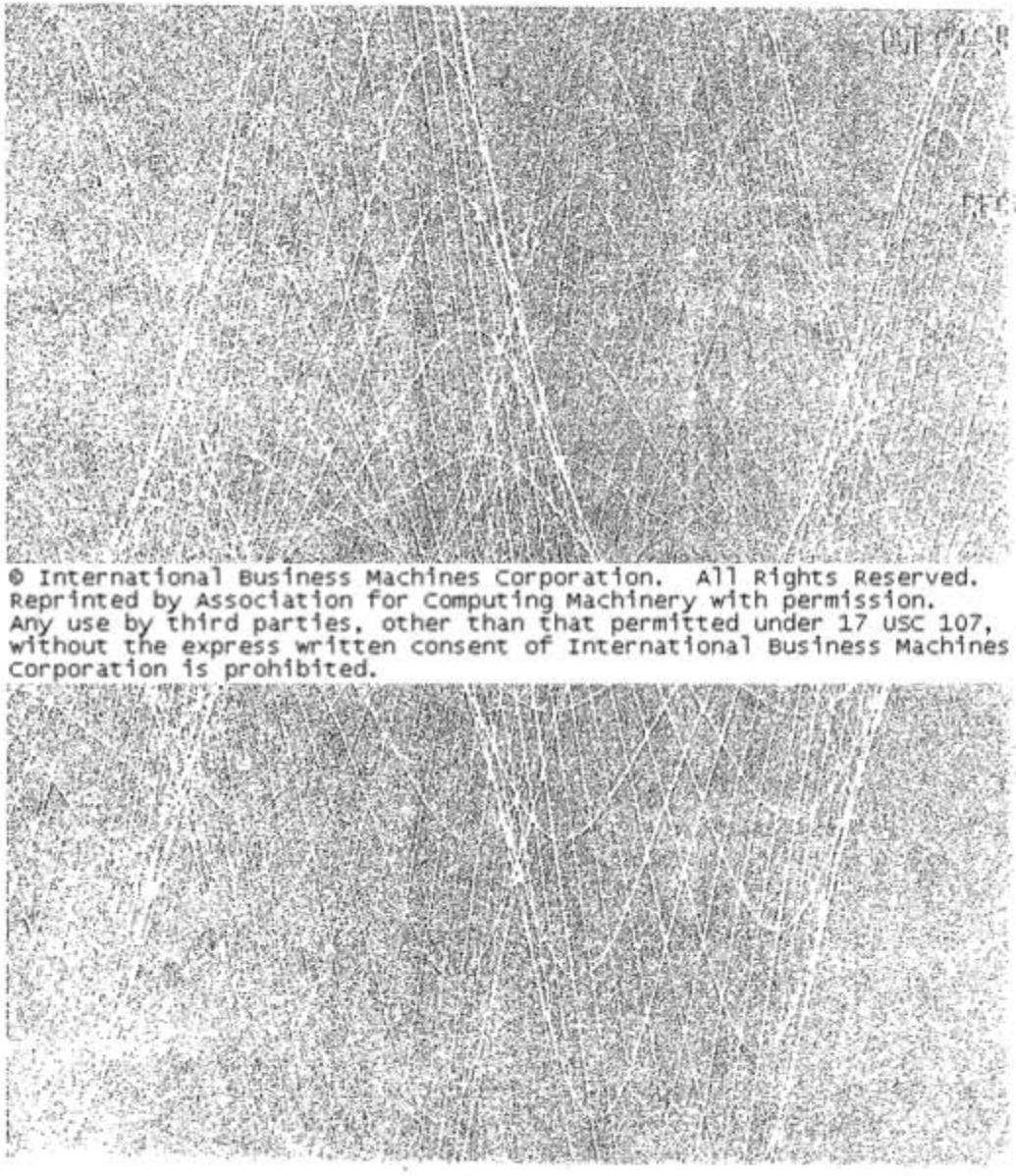
- Program Committee Member: ICDE, PODS, SIGMOD, VLDB, etc.
- CIKM 2008 Track Chair

Awards received:

- NSF Career Award, 2004
- IBM Pat Goldberg Memorial Best Paper Award, 2004
- Best Paper Award for Computer Networks Special Issue on XML, 2002

Statement:

SIGMOD/PODS conference and its affiliated workshops attract database researchers, practitioners, and students from all corners of the world every year. I am honored to be nominated and eager to do my best to serve ACM SIGMOD on the Executive Committee. If elected as Treasurer, I will continue to ensure healthy budgets for SIGMOD/PODS conference in the years to come, and do my best in the SIGMOD Executive Committee to ensure successful outcomes for its endeavors.



2

RJ 599

DERIVABILITY, REDUNDANCY AND CONSISTENCY OF RELATIONS STORED IN LARGE DATA BANKS

E. F. Codd

August 19, 1969

IBM RESEARCH

DERIVABILITY, REDUNDANCY AND CONSISTENCY OF RELATIONS STORED IN LARGE DATA BANKS

E. F. Codd
Research Division
San Jose, California

ABSTRACT: The large, integrated data banks of the future will contain many relations of various degrees in stored form. It will not be unusual for this set of stored relations to be redundant. Two types of redundancy are defined and discussed. One type may be employed to improve accessibility of certain kinds of information which happen to be in great demand. When either type of redundancy exists, those responsible for control of the data bank should know about it and have some means of detecting any "logical" inconsistencies in the total set of stored relations. Consistency checking might be helpful in tracking down unauthorized (and possibly fraudulent) changes in the data bank contents.

RJ 599 (# 12343)
August 19, 1969

CONTENTS

SIGMOD Record, March 2009 (Vol. 38, No. 1)

INTRODUCTION

The first part of this paper is concerned with an explanation of a relational view of data. This view (or model) of data appears to be superior in several respects to the graph or network model [1, 2] presently in vogue. It provides a means of describing data with its natural structure only: that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level retrieval language which will yield maximal independence between programs on the one hand, and machine representation and organization of data on the other. A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations -- these are discussed in the second part of this paper. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations. Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present management information systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system.

1. A Relational View of Data

The term relation is used here in its accepted mathematical sense. Given sets S_1, S_2, ..., S_n (not necessarily distinct), R is a relation on these n sets if it is a set of n-tuples, each of which has its first element from S_1, its second element from S_2, and so on. We shall refer to S_j as the j^{th} domain of R. As defined above, R is said to have degree n. Relations of degree 1 are often called unary, degree 2 binary, degree 3 ternary, and degree n n-ary.

For expository reasons, we shall frequently make use of an array representation of relations, but it must be remembered that this particular representation is not an essential part of the relational view being expounded. An array which represents an n-ary relation R has the following properties:

(1) Each row represents an n-tuple of R;

(2) The ordering of rows is immaterial;

(3) All rows are distinct;

(4) The ordering of columns is significant - it corresponds to the ordering S_1, S_2, ..., S_n of the domains on which R is defined;

(5) The significance of each column is partially conveyed by labeling it with the name of the corresponding domain.

The example in Figure 1 illustrates a relation of degree 4 called ship which reflects the shipments-in-progress of parts from specified suppliers to specified projects in specified quantities.

ship (supplier	part	project	quantity)
1	2	5	17
1	3	5	23
2	3	7	9
2	7	5	4
4	1	1	12

FIGURE 1: A Relation of Degree 4

One might ask: If the columns are labeled by the name of the corresponding domains, why should the ordering of columns matter? As the example in Figure 2 shows, two columns may have identical headings (indicating identical domains), but possess distinct meanings with respect to the relation. The relation depicted is called component. It is a binary relation, each of whose two domains is called part. The meaning of component (x, y) is that part x is an immediate component (or subassembly) of part y.

component (part	part)
1	5
2	5
3	5
2	6
3	6
4	7
6	7

Figure 2: A Relation with Two Identical Domains

We now assert that a data bank is a collection of time-varying relations. These relations are of assorted degrees. As time progresses, each n-ary relation may be subject to insertion of additional n-tuples, deletion of existing ones, and alteration of components of any of its existing n-tuples.

Consider, for example, a data bank which contains information about parts, projects, and suppliers. The individual description for an individual object (such as a particular part) is called an entity [3]. The prototype description for a class of objects is called an entity type. The set of entities of a given entity type can be viewed as a relation, and we shall call such a relation an entity type relation. In the example under consideration, there might be an entity type relation called part defined on the following domains:

(1) part number

(2) part name

(3) part color

(4) part weight

(5) quantity on hand

(6) quantity on order

and possibly other domains as well. Each of these domains is, in effect, a pool of values, some or all of which may be represented in the data bank at any instant. While it is conceivable that, at some instant, all part colors are present, it is unlikely that all possible part weights, part names, and part numbers are. The domains listed above correspond to what are commonly called the attributes of the entity type part.

Normally, one attribute (or combination of attributes) of a given entity type has values which uniquely identify each entity. Such an attribute (or combination) is called a key. In the example above, part number would be a key, while part color would not be. A key is non-redundant if it is either a simple attribute (not a combination) or a combination such that none of the participating attributes is superfluous in uniquely identifying each entity. An entity type may possess more than one non-redundant key. This would be the case in the example, if different parts were always given distinct names.

The remaining relations in a data bank are between entity types, and are, therefore, called inter-entity relations. An essential property of every inter-entity relation is that its domains include at least two keys which either refer to distinct entity types or refer to a common entity type serving distinct roles.

The examples in Figures 1 and 2 will help clarify this. The relation exhibited in Figure 1 involves three keys, one for each of the entity types supplier, part, project. The relation exhibited in Figure 2 involves two keys referring to the common entity type part, the first key serving to identify a component, the second to identify an assembly containing that component. Both of these relations are inter-entity relations.

So far, we have discussed examples of relations which are defined on simple domains - domains whose elements are atomic (non-decomposable) values. Non-atomic values can be discussed within the relational framework. Thus, some domains may have relations as elements. These relations may, in turn, be defined on non-simple domains, and so on. For example, one of the domains on which the entity type relation employee is defined might be salary history. An element of the salary history domain is a binary relation defined on the domain date and the domain salary. The salary history domain is the set of all such binary relations.

2. Some Linguistic Aspects

The adoption of a relational view of data, as described above, permits the development of a universal retrieval sublanguage based on the second-order predicate calculus.* Such a language would provide a yardstick of linguistic power for all other proposed retrieval languages, and would itself be a strong candidate for embedding (with appropriate syntactic modification) in a variety of host languages (programming, command or problem oriented). While it is not the purpose of this paper to describe such a language in detail, its salient features would be as follows.

Let us denote the retrieval sublanguage by R and the host language by H. R permits the declaration of domains, together with relations of various degrees on those domains. H permits supporting declarations which indicate, perhaps less permanently, how these relations are represented in storage. R permits the specification for retrieval of any subset of data from the data bank. Action on such a retrieval request is subject to security constraints.

The class of qualification expressions which can be used in a set specification is in a precisely specified one-to-one correspondence with the class of well-formed formulas of the predicate calculus in prenex normal form [4]. Any arithmetic functions needed can be defined in H and invoked in R. A set so specified may be fetched for query purposes only or it may be held for possible changes. Insertions take the form of adding new elements to declared relations without regard to any ordering that may be present in their machine representation. Deletions which are effective for the community (as opposed to the individual user or sub-communities) take the form of removing elements from declared relations. Some deletions may be triggered by others, if deletion dependencies between specified relations are declared in R.

One important effect that the view adopted toward data has on the language used to retrieve it is in the naming of data elements and sets. With the usual network view, users will often be burdened with coining and using more relation names than are absolutely necessary, since names are associated with directed paths rather than with relations. Two such paths are needed to support symmetric exploitation* of a single binary relation. For a relation of degree n, the number of paths to be named and controlled is factorial n.

Again, if a relational view is adopted in which every n-ary relation (n > 2) has to be expressed by the user as a nested

*The second-order predicate calculus (rather than first-order) is needed because the domains on which relations are defined may themselves have relations as elements (see section 1).

*Once a user is aware that a certain relation is stored, he will expect to be able to exploit it using any combination of its arguments as "knowns" and the remaining arguments as "unknowns," because the information (like Everest) is there. This is a system feature (missing from many current information systems), which we shall call (logically) symmetric exploitation of relations.

SIGMOD Record, March 2009 (Vol. 38, No. 1)

expression involving only binary relations, then 2n-1 names have to be coined instead of only n+1 with direct n-ary notation as described in Section 1. For example, the 4-ary relation ship of Figure 1, which entails 5 names in n-ary notation, would be represented in the form

P (supplier, Q (part, R (project, quantity)))

in nested binary notation and, thus, employ 7 names.

3. Operations on Relations

Since relations are sets, all of the usual set operations are applicable to them. Nevertheless, the result may not be a relation; for example, the union of a binary relation and a ternary relation is not a relation.

The operations discussed below are specifically for relations. These operations are introduced because of their key role in deriving relations from other relations. Most users would not be directly concerned with these operations. Information systems designers and people concerned with data bank control should, however, be thoroughly familiar with these operations.

3.1 Permutation

A binary relation has an array representation with two columns. Interchanging these two columns yields the converse relation. More generally, if a permutation is applied to the columns of an n-ary relation, the resulting relation is said to be a permutation of the given relation. There are, for example, 4! = 24 permutations of the relation ship in Figure 1, if we include the identity permutation which leaves the ordering of columns unchanged.

In a system which provides symmetric exploitation of relations, the set of queries answerable by a stored relation is identical to the set answerable by any permutation of that relation. Although it is logically unnecessary to store both a relation and some permutation of it, performance considerations could make it advisable.

3.2 Projection

Suppose now we select certain columns of a relation (striking out the others) and then remove from the resulting array any duplication in the rows. The final array represents a relation which is said to be a projection of the given relation.

A selection operator Π is used to obtain any desired permutation, projection, or combination of the two operations. Thus, if L is a list of k indices

$$L = i_1, i_2, \ldots, i_k$$

and R is an n-ary relation ($n \geq k$), then $\Pi_L(R)$ is the k-ary relation whose j^{th} column is column i_j of R ($j = 1, 2, \ldots, k$) except that duplication in resulting rows is removed. Consider

the relation ship of Figure 1. A projection of this relation is exhibited in Figure 3.

Π_{31} (ship)

(project	supplier)
5	1
5	2
1	4
7	2

Figure 3: A Permuted Projection of the Relation in Figure 1

Note that, in this particular case, the projection has fewer n-tuples than the relation from which it is derived.

3.3 Join

Suppose we are given two binary relations, which have some domain in common. Under what circumstances can we combine these relations to form a ternary relation which preserves all of the information in the given relations?

The example in Figure 4 shows two relations R, S, which are joinable without loss of information, while Figure 5 shows a join of R with S. A binary relation R is joinable with a binary relation S if there exists a ternary relation U such that $\Pi_{12}(U) = R$ and $\Pi_{23}(U) = S$. Any such ternary relation is called a join of R with S. If R, S are binary relations such that $\Pi_2(R) = \Pi_1(S)$, then R is joinable with S. One join that always exists in such a case is the natural join of R with S defined by

$$R*S = \{(a, b, c):R(a, b) \wedge S(b, c)\}$$

where R(a, b) has the value true if (a, b) is a member of R and similarly for S(b, c). It is immediate that

$$\Pi_{12}(R*S) = R$$

and

$$\Pi_{23}(R*S) = S$$

Note that the join shown in Figure 5 is the natural join of R with S from Figure 4. However, this join is not the only one of R with S. Figure 6 shows another possible join of the relations in Figure 4.

R (supplier	part)	S (part	project)
1	1	1	1
2	1	1	2
2	2	2	1

Figure 4: Two Joinable Relations

SIGMOD Record, March 2009 (Vol. 38, No. 1)

R*S (supplier	part	project)
1	1	1
1	1	2
2	1	1
2	1	2
2	2	1

Figure 5: The Natural Join of R with S (from Figure 4)

U (supplier	part	project)
1	1	2
2	1	1
2	2	1

Figure 6: Another Join of R with S (from Figure 4)

Inspection of these relations reveals an element (element 1) of the domain part (the domain on which the join is to be made) with the property that it possesses more than one relative under R and also under S. It is this element which gives rise to the plurality of joins. Such an element in the joining domain is called a point of ambiguity with respect to the joining of R with S.

If either $\Pi_{21}(R)$ or S is a function*, no point of ambiguity can occur in joining R with S. In such a case, the natural join of R with S is the only join of R with S. Note that the reiterated qualification "of R with S" is necessary, because S might be joinable with R (as well as R with S), and this join would be an entirely separate consideration. In Figure 4, none of the relations R, $\Pi_{21}(R)$, S, $\Pi_{21}(S)$ is a function.

Ambiguity in the joining of R with S can sometimes be resolved by means of other relations. Suppose we are given, or can derive from sources independent of R and S, a relation T on the domains project and supplier with the following properties:

(1) $\Pi_1(T) = \Pi_2(S)$

(2) $\Pi_2(T) = \Pi_1(R)$

(3) $T(j, s) \rightarrow \exists p(R(s, p) \wedge S(p, j))$

(4) $R(s, p) \rightarrow \exists j(S(p, j) \wedge T(j, s))$

(5) $S(p, j) \rightarrow \exists s(T(j, s) \wedge R(s, p))$,

then we may form a three-way join of R, S, T; that is, a ternary relation such that

$$\Pi_{12}(U) = R$$

$$\Pi_{23}(U) = S$$

$$\Pi_{31}(U) = T.$$

*A function is a many-one binary relation.

Such a join will be called a cyclic 3-join to distinguish it from a linear 3-join which would be a quaternary relation V such that

$$\Pi_{12}(V) = R$$

$$\Pi_{23}(V) = S$$

$$\Pi_{34}(V) = T.$$

While it is possible for more than one cyclic 3-join to exist (see Figures 7, 8 for an example), the circumstances under which this can occur entail much more severe constraints than those for a plurality of 2-joins. To be specific, the relations R, S, T must possess points of ambiguity with respect to joining R with S (say point x), S with T (say y), and T with R (say z), and, furthermore, y must be a relative of x under S, z a relative of y under T, and x a relative of z under R. Note that in Figure 7 the points

$$x = a; \quad y = d; \quad z = 2$$

have this property.

R (s	p)
1	a
2	a
2	b

S (p	j)
a	d
a	e
b	d
b	e

T (j	s)
d	1
d	2
e	2

Figure 7: Binary Relations with a Plurality of Cyclic 3-Joins

U (s	p	j)
1	a	d
2	a	e
2	b	d
2	b	e

U' (s	p	j)
1	a	d
2	a	d
2	a	e
2	b	d
2	b	e

Figure 8: Two Cyclic 3-Joins of the Relations in Figure 7

The natural linear 3-join of three binary relations R, S, T is given by

$$R*S*T = \{(a, b, c, d):R(a, b) \wedge S(b, c) \wedge T(c, d)\}$$

where parentheses are not needed on the left hand side because the natural 2-join (*) is associative. To obtain the cyclic counterpart, we introduce the operator γ which produces a relation of degree n-1 from a relation of degree n by tying its ends together. Thus, if R is an n-ary relation

$$\gamma(R) = \{(a_1, a_2, \ldots, a_{n-1}):R(a_1, a_2, \ldots, a_{n-1}, a_n) \wedge a_1 = a_n\}.$$

We may now represent the natural cyclic 3-join of R, S, T by the expression

$$\gamma(R*S*T).$$

SIGMOD Record, March 2009 (Vol. 38, No. 1)

Extension of the notions of linear and cyclic 3-join and their natural counterparts to the joining of n binary relations (where $n \geq 3$) is obvious. A few words may be appropriate, however, regarding the joining of relations which are not necessarily binary. Consider the case of two relations R (degree r), S (degree s) which are to be joined on p of their domains ($p < r$, $p < s$). For simplicity, suppose these p domains are the last p of the r domains of R, and the first p of the s domains of S. If this were not so, we could always apply appropriate permutations to make it so. Now, take the cartesian product of the first r-p domains of R, and call this new domain A. Take the cartesian product of the last p domains of R, and call this B. Take the cartesian product of the last s-p domains of S and call this C.

We can treat R as if it were a binary relation on the domains A, B. Similarly, we can treat S as if it were a binary relation on the domains B, C. The notions of linear and cyclic 3-join are now directly applicable. A similar approach can be taken with the linear and cyclic n-joins of n relations of assorted degrees.

3.4 Composition

The reader is probably familiar with the notion of composition applied to functions. We shall discuss a generalization of that concept and apply it first to binary relations. Our definitions of composition and composability are based very directly on the definitions of join and joinability given above.

Suppose we are given two relations R, S. T is a composition of R with S if there exists a join U of R with S such that $T = \Pi_{13}(U)$. Thus, two relations are composable if and only if they are joinable. However, the existence of more than one join of R with S does not imply the existence of more than one composition of R with S.

Corresponding to the natural join of R with S is the natural composition of R with S defined by

$$R \cdot S = \Pi_{13}(R * S).$$

Taking the relations R, S from Figure 4, their natural composition is exhibited in Figure 9 and another composition is exhibited in Figure 10 (derived from the join exhibited in Figure 6).

R·S (project	supplier)
1	1
1	2
2	1
2	2

Figure 9: The Natural Composition of R with S (from Figure 4)

T (project	supplier)
1	2
2	1

Figure 10: Another Composition of R with S (from Figure 4)

When two or more joins exist, the number of distinct compositions may be as few as one or as many as the number of distinct joins. Figure 11 shows an example of two relations which have several joins but only one composition. Note that the ambiguity of point c is lost in composing R with S, because of unambiguous associations made via the points a, b, d, e.

R (supplier	part)
1	a
1	b
1	c
2	c
2	d
2	e

S (part	project)
a	g
b	f
c	f
c	g
d	g
e	f

Figure 11: Many Joins, Only One Composition

Extension of composition to pairs of relations which are not necessarily binary (and which may be of different degrees) follows the same pattern as extension of pairwise joining to such relations. We now proceed to make use of these operations on relations in considering what relations need to be actually stored.

4. Expressible, Named and Stored Relations

Associated with a data bank are three collections of relations:

(1) the expressible set

(2) the named set

(3) the stored set

The expressible set is the collection of relations which can be designated by expressions in the retrieval language for the purpose of defining sets of data to be retrieved. Such expressions are constructed from simple names of relations, relational operators such as =, logical connectives and the quantifiers of the predicate calculus.

The named set is the collection of all relations in the data bank which the user can identify by means of simple public names. This set is a subset of the expressible set - usually a very small subset.

The stored set is the collection of all relations whose values are actually stored in the data bank. This set would normally be a subset of the named set, and we assume that it is. If the traffic on some unnamed but expressible relation grows to such proportions that such a relation should be included in the stored set, then it should be given a public name and thereby included in the named set.

SIGMOD Record, March 2009 (Vol. 38, No. 1)

Those relations which are in the named set and not in the stored set are defined by expressions (independent of time) involving names of relations in the stored set, together with the permutation-projection, natural composition, natural join and tie operators (Π, $\cdot$, $*$, γ). Such definitions by expressions must be within the scope of the retrieval language R.

Decisions regarding which relations belong in the named set are based mainly on the logical needs of the community of users, and particularly on the ever-increasing investment in programs using relations by name as a result of past membership of these relations in the named set. On the other hand, decisions regarding which relations belong in the stored set are based mainly on the transaction and interaction loads, the performance requirements of the users, and changes that take place in these factors.

5. Derivability, Redundancy and Consistency

A relation R is derivable from a set S of relations if there exists a sequence* of permutations, projections, natural joins, and ties which yields R from members of S. This sequence of operations yields a correct value for R at virtually any time (for the stored set, we must exclude times at which changes are actually being made to the values of R and S). Note that, because natural join is specified, there is no question as to which join to take.

A set of relations is strongly redundant if it contains at least one relation which is derivable from the rest of the members. While the named set of relations is likely to be redundant in this sense for user convenience, the stored set will often be non-strongly-redundant in order to save storage space as well as time to perform updates, insertions, and deletions. Only in an environment with a heavy load of queries relative to the other kinds of interaction with the data bank would strong redundancy be justified in the stored set of relations.

A set of relations is weakly redundant if it contains at least one relation which is not derivable from other members of the set, but is at all times a projection of some join of other members of the set. The join in question might be the natural one at some time and an unnatural one at some other time.

Generally speaking, weak redundancies are inherent in the logical needs of the community of users. They are not removable by the system or data base administrator. If they appear at all, they appear in both the named and stored sets. Strong redundancies, on the other hand, are removable from the stored set, providing the resulting performance changes are acceptable.

*We can omit natural composition in the list of operations, because it is a combination of a join and a projection.

As an example of a weak redundancy, consider the case cited previously in which there are binary relations R, S, T with meanings as follows:

R(s, p) supplier s supplies part p to at least one project

S(p, j) part p is supplied by at least one supplier to project j

T(j, s) project j is supplied at least one kind of part by supplier s

All three relations are complex* relations with the possibility of points of ambiguity occurring from time to time in the potential joining of any two. Hence, none of them is derivable from the other two. However, constraints do exist between them, since each is a projection of some cyclic join of the three of them. Thus, this set of relations possesses a weak redundancy.

Whenever a set of relations is redundant in either sense, we shall associate with that set a collection of statements which define all of the redundancies which hold independent of time between the member relations. If the information system lacks - and it most probably will - detailed semantic information about each named relation, it cannot deduce the redundancies applicable to the named set. It might, over a period of time, make attempts to induce the redundancies, but such attempts would be fallible.

Given a collection C of relations and an associated set of constraint statements, we shall call C consistent or inconsistent according as it does or does not comply with the stated redundancies. For example, given stored relations R, S, T together with the constraint statement

"$\Pi_{12}(T)$ is a composition of $\Pi_{12}(R)$ with $\Pi_{12}(S)$",

we may check from time to time that the values stored for R, S, T satisfy this constraint. An algorithm for making this check would examine the first two columns of each of R, S, T (in whatever way they are represented in the system) and determine whether

(1) $\Pi_1(T) = \Pi_1(R)$

(2) $\Pi_2(T) = \Pi_2(S)$

(3) for every element pair (a, c) in the relation $\Pi_{12}(T)$ there is an element b such that (a, b) is in $\Pi_{12}(R)$ and (b, c) is in $\Pi_{12}(S)$.

There are practical problems (which we shall not discuss here) in taking an instantaneous snapshot of a collection of relations, some of which may be very large and highly variable.

*A binary relation is complex if neither it nor its converse is a function.

6. Data Bank Control

Inconsistencies in a collection of relations may arise due to inadequate or faulty input. An example of inadequate input is the insertion of a new element, say (2, 5) in the relation S (part, project) when part 2 has no supplier who supplies project 5 (see previous section). The generation of an inconsistency of this kind could be logged internally, so that if it were not remedied within some reasonable time interval by appropriate insertions in the relations R, T the system could notify the security officer. Alternatively, the system could assist the user in making insertions and deletions by informing him that such and such relations now need to be changed to restore consistency in the collection. Ideally, it should be possible to make different selections of system reaction to inconsistency for different subcollections of relations in an individual data bank.

ACKNOWLEDGMENT

The author wishes to thank Dr. F. P. Palermo and Dr. E. B. Altman of the San Jose Research Laboratory for helpful discussions.

REFERENCES

1. C. W. Bachman, "Software for Random Access Processing," Datamation, April 1965.

2. W. C. McGee, "Generalized File Processing," Annual Review in Automatic Programming 5, 13, pp. 77-149, Pergamon Press, 1969.

3. G. H. Mealy, "Another Look at Data," Proceedings Fall Joint Computer Conference, 1967.

4. A. Church, "An Introduction to Mathematical Logic I," Princeton, 1956.

An X-Ray on Web-Available XML Schemas

Alberto H. F. Laender, Mirella M. Moro, Cristiano Nascimento and Patrícia Martins*
Department of Computer Science
Federal University of Minas Gerais
Belo Horizonte, Brazil
{laender, mirella, crist}@dcc.ufmg.br, patricia.martins@gmail.com

ABSTRACT

XML has conquered its place as the most used standard for representing Web data. An XML schema may be employed for similar purposes of those from database schemas. There are different languages to write an XML schema, such as DTD and XSD. In this paper, we provide a general view, an X-Ray, on Web-available XSD files by identifying which XSD constructs are more and less frequently used. Furthermore, we provide an evolution perspective, showing results from XSD files collected in 2005 and 2008. Hence, we can also draw some conclusions on what trends seem to exist in XSD usage. The results of such study provide relevant information for developers of XML applications, tools and algorithms in which the schema has a distinguished role.

1. INTRODUCTION

A large volume of data is currently represented as XML documents [8]. With such a widespread use, XML has conquered its place as the most employed standard for representing Web data. Web applications frequently need to specify a schema (or a set of schemas) to their documents in order to fulfill their requirements. An XML schema defines a class of documents, i.e., the constraints that all documents must follow in order to be valid. As the complexity of the applications grow, so does the complexity of their schemas.

There are different languages to write an XML schema [12], such as Document Type Definition (DTD) [1] and XML Schema [16]. An XML schema may be employed for similar purposes of those from database schemas. For example, it defines the data structure, extracting information about the data organization, which in turn is essential to store the data in a DBMS. Also, a query optimizer may use facts from the schema for improving its performance. Without such an information, it needs to infer a schema from a dataset, which is a different problem [14]. Finally, other research fronts, such as information integration [5], schema evolution [13], and web services [19], may also benefit from knowing how exactly the schema definitions are used in the real world.

Considering all those schema-oriented applications, some previous work has presented how real DTDs and XML Schema Definitions (XSD) look like. One of the first ones is [10], which provides statistics about the structure of Web-collected DTDs. Another study compared the power expression of real DTDs and XSDs [6]. It concluded that, considering power expression only, most of the XSDs could be written as a similar DTD file. Then, the authors in [3] studied how publicly available XML documents are. They presented statistics on document distribution, schema usage, document internal features, among others.

The goal of the present paper is to provide a general view, an X-Ray, on Web-available XSD files. Specifically, we want to identify which XSD constructs are more and less frequently used. Also, we provide an evolution perspective, showing results from XSD files collected in March 2005 and in November 2008. We can also draw some conclusions on what trends seem to exist in XSD usage. The results of such study provide relevant information for developers of XML applications and tools in which the schema has a defining role.

In summary, this paper presents a snapshot of Web-available XSDs. It works like the initial X-Ray a doctor ask for a patient, trying to grasp a general view of a problem. As the doctor may request an MRI or a CT for enhanced investigation, so can we. For example, we could further examine XSDs by performing an application-based clustering in which we would group XSDs related to bio-data, math-data, and so on. Then, we could evaluate how the frequency of XSD constructs vary from one to another cluster. We leave this deeper study of XSDs for future work, since we are interested in a broader perspective with more general purposes. Finally, as an X-Ray may have different interpretations, so can our study. Hence, we do not exhaustively interpret our results, leaving them open for further discussion as well.

This paper is organized as follows. Section 2 summarizes some concepts about XML and XML Schema. Section 3 presents the methodology employed to perform our evaluations. The results of our experiments are presented and discussed in Section 4. Section 5 overviews some related work while Section 6 concludes the paper.

2. BACKGROUND

XML (*Extensible Markup Language*) is a meta-markup language that provides a format to describe structured and semi-structured data [1]. XML enables more precise content definition and more efficient document search, working over multiple plataforms. Moreover, it also allows the definition

*Research partially supported by projects InfoWeb (CNPq grant number 573871/2008-6) and Amanaje (CNPq grant number 479541/2008-6), Brazil.

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<!DOCTYPE SigmodRecord SYSTEM SigmodRecord.dtd">
<SigmodRecord>
 <issue>
  <volume>11</volume>
  <number>1</number>
  <articles>
    <article>
    <title>Annotated Bibliography on Data Design.</title>
    <initPage>45</initPage>
    <endPage>77</endPage>
   <authors>
    <author position="00">Anthony I. Wasserman</author>
    <author position="01">Karen Botnich</author>
   </authors>
  </article>
  <article>
   <title>Architecture of Future Data Base Systems.</title>
   <initPage>30</initPage>
   <endPage>44</endPage>
   <authors>
    <author position="00">Lawrence A. Rowe</author>
    <author position="01">Michael Stonebraker</author>
   </authors>
  </article>
  ...
 </issue>
</SigmodRecord>
```

Figure 1: XML document for SIGMOD Record

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      elementFormDefault="qualified">
 <xs:element name="SigmodRecord">
  <xs:complexType>
   <xs:sequence>
    <xs:element maxOccurs="unbounded" ref="issue"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="issue">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="volume"/>
    <xs:element ref="number"/>
    <xs:element ref="articles"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 ...
 <xs:element name="author">
  <xs:complexType mixed="true">
   <xs:attribute name="position" use="required"
                                        type="xs:integer"/>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Figure 2: An XSD file for SIGMOD Record

of a new generation of applications to handle and visualize Web data. While its counterpart HTML has a fixed set of tags to define the format of characters and paragraphs, XML provides a system to define an *infinite* number of tags (markups). Specifically, an XML document is composed of element definitions and text values. For example, the XML document for the SIGMOD Record[1] issues is composed of elements that define each issue with volume and number, its articles, and the information of each article (title, pages, and authors). Figure 1 shows a sample of this document.

An XML schema may be defined to guarantee that each document of an application or domain follows the same structural constraints. The most common schema languages are DTD [1] and XML Schema [16]. This paper focuses on analyzing how the actual XSD files are used on the Web. Therefore, this section briefly overviews some XML Schema definition features.

An XSD file is an XML document that allows to define: the elements and attributes that may appear in an XML document, the order and the number of child elements, whether the element is empty or has text content, the data types for elements and attributes, default and pattern values. For example, Figure 2 illustrates a part of an XSD file for the SIGMOD Record document from Figure 1.

Note that this is *one* option for the schema definition of the SIGMOD Record document, a very generic XSD file. We could make it more specific by adding some constraints. For example, SIGMOD Record has four issues per year. We could include such restriction by specifying the element *number* as a *simpleType* with constraints *minInclusive* equal to 1 and *maxInclusive* equal to 4.

Furthermore, elements may be specified as simple types (such as *string* and *positiveInteger*) or complex types. Complex types allow more richer specifications. For example, we can use *group* to define a set of elements. Then, we can add one of three restrictions: (i) *sequence*, those elements must appear in the document in the sequence that they are defined in the XSD; (ii) *choice*, only one of those elements may appear in the document; (iii) *all*, either all elements appear (in any order) or none of them appear in the document.

3. EVALUATION SETUP

This section overviews our evaluation methodology. The process has three steps: it gets data from the Web, validates them, and parses the validated data.

1. Get XSD files from the Web. We consider XML schemas available on the Web for the following reasons. First, those schemas are publicly available, making it easier to reproduce our evaluation. Second, they represent currently in-use XSD files, providing a good snapshot of the real world scenario. Third, those files are not specialized in a limited set of fields (e.g., bio or math data) such as those found in some XSD repositories.

The schemas were selected through a search engine - in our case we have used Google[2]. The query considered on the search engine was "*schema filetype:xsd*". With such a query, we expected to get links only to XSD files that instantiate an element *schema*.

[1]http://www.sigmod.org/record

[2]http://www.google.com, last access in November 2008.

Next task is to get the actual XSD files. We have implemented a crawler using Perl facilities (an HTML parser[3] and a library for WWW in Perl[4]). The crawler gets the URLs that are on the resulting pages of the query "*schema filetype:xsd*" (on the search engine). We have considered the URLs from the first 50 pages, where each page has 10 results. The crawler then downloads the XSD files from those URL links. At the end of this step, we had approximately 500 XSD files, since some of URL links were broken.

2. Validate retrieved XSD files. Once we have a dataset of XSD files, we needed to validate them. We have used a Java Validation API for XML documents[5], and validated schemas against XML Schema's schema[6]. We performed our evaluation in March 2005 and again in November 2008. From the 500 URL links, we got 199 valid XSD files in 2005, and 223 valid XSD files in 2008.

3. Parse validated XSD files. In order to get our statistics, we have also implemented a parser that captures the specific information we are interested in. Specifically, we have defined a script that uses a Perl XML DOM API[7] to parse documents. Its input is a list of XSD files, and its output is the frequency of each XML schema construct. Finally, we have not considered XSD files with *namespace*, *include* or *import* constructs, because we do not have control of them.

4. RESULTS

In this paper, we apply two strategies to evaluate the frequencies in which the XSD constructs appear in the files. The first one, called *general usage* (GEN), counts how many **documents** (from the XSD files dataset) use each construct at least once. This measure tells us which constructs are more/less frequent within the **set** of documents. The second one, called *internal usage* (INT), measures the frequency of each construct over all constructs within each **single** document. In other words, it evaluates how important certain constructs are for each document.

4.1 General Overview

We start our evaluation by presenting a general overview of the constructs considered in Table 1. The first column lists the name of the construct being evaluated. The second and third columns list the general usage on both datasets from 2005 and 2008. For example, the construct *all* appears in 10.55% of all XSD files collected in 2005, and in 19.82% of those collected in 2008. Finally, the last two columns list the internal usage from the same datasets. For example, the construct *all* represents 0.36% of all constructs used within one single document, considering the average of all 2005 files, and in 0.49% of all 2008 files.

Note that GEN represents the frequency in which each construct appears in the overall dataset. On the other hand, INT reflects the internal structure of the XSD files, individually. Therefore, we analyze those results separetely.

[3]http://search.cpan.org/dist/HTML-Parser
[4]http://search.cpan.org/~gaas/libwww-perl-5.800
[5]http://java.sun.com/j2se/1.5.0/docs/api/javax/xml/validation/package-summary.html
[6]http://www.w3.org/2001/XMLSchema.xsd
[7]http://search.cpan.org/dist/XML-DOM

Table 1: Usage of XML Schema elements (%)

Construct	GEN-05	GEN-08	INT-05	INT-08
all	10.55	19.82	0.36	0.49
annotation	59.80	81.98	8.79	10.37
any	15.07	31.53	0.08	0.33
anyAttribute	6.53	10.81	0.03	0.19
appinfo	5.53	9.91	0.08	0.09
attribute	79.40	81.08	7.89	9.34
attributeGroup	8.54	11.71	1.01	0.60
choice	46.23	42.34	0.94	0.84
complexContent	19.09	30.63	0.71	1.38
complexType	91.46	92.79	5.77	8.83
documentation	58.80	81.98	9.40	10.59
element	92.96	93.69	25.66	26.57
enumeration	56.78	72.97	19.35	8.68
extension	47.24	53.15	1.46	2.04
field	10.05	16.22	1.40	0.45
fractionDigits	2.51	3.60	0.16	0.04
group	8.04	12.61	0.47	0.55
key	7.54	9.91	0.29	0.12
keyref	5.53	7.21	0.27	0.13
length	4.52	8.11	0.02	0.04
list	5.53	7.21	0.04	0.06
maxExclusive	2.51	4.50	0.04	0.01
maxInclusive	15.07	28.83	0.40	0.35
maxLength	11.56	20.72	0.52	0.42
minExclusive	2.51	6.31	0.27	0.03
minInclusive	17.59	32.43	0.41	0.35
minLength	11.56	20.72	0.14	0.41
notation	0.50	0.00	0.00	0.00
pattern	24.62	36.04	0.93	1.14
restriction	68.34	82.88	3.31	3.44
selector	10.05	16.22	0.89	0.42
sequence	87.94	89.19	3.50	5.79
simpleContent	36.18	37.84	0.76	0.72
simpleType	68.84	82.88	3.66	3.63
totalDigits	2.51	3.60	0.17	0.05
union	5.53	15.32	0.31	0.19
unique	5.53	13.51	0.32	0.17
whiteSpace	4.02	4.50	0.07	0.09

General Usage

We now discuss some of the most interesting results presented in the first three columns of Table 1. The constructs that are most used are: *complexType*, *element*, *sequence*, *simpleType*, and *restriction*. The least used ones are *notation*, *totalDigits*, *fractionDigits*, *maxExclusive*, and *whiteSpace*. There is an interesting difference between those two sets: the former is more related to the document structure, while the latter to the values inside the elements.

It is important to notice that there is a general increase in the use of all constructs over the last three years. The exceptions are the constructs *choice* and *notation*. Such an increase appears in the most frequently used constructs and in the least ones as well. Also, the most frequent ones (i.e., *element*, *complexType*, and *attribute*) have increased less in relation to the others. This can indicate the stability on the use of such constructs.

Considering the constructs that had a high increase from 2005 to 2008 (such as *documentation*, *annotation*, *any*, *enu-*

Table 2: Content model distribution in complexType construct

Content Model	Internal Usage	
	2005 (%)	2008 (%)
Simple	13.11	13.45
Complex	86.89	86.55

meration, *minInclusive*, and *restriction*), one possible explanation would be that the users are more familiar with them now. Also, as more tools for handling XML schemas become available, the users start to acquire confidence in using more complex structures as well. Moreover, as XML applications become more specific, so do their requirements and data restrictions. For example, instead of using an element of type *xs:int*, it may be necessary to employ a *simpleType* with restrictions in its minimum and maximum values. Particularly expressive is the use of the construct *documentation*, which occurred in 53.80% of the 2005 XSD files and in 81.92% of the 2008 ones. This is a clear indication that XSDs are, in general, very complex and need to be properly documented.

Another explanation for the general increasing trend would be the cascade effect. In other words, the increase of using one construct may have affected the increasing of others. Such a consequence is expected because, most of the times, the constructs allow nesting. Therefore, if the use of an external construct increases, the same will happen to its internal constructs. For example, the constructs *maxExclusive*, *maxInclusive*, *minExclusive*, *minExclusive*, *restriction*, *pattern*, *enumeration*, *totalDigits*, and *fractionDigits* are usually associated to the construct *simpleType*. Hence, *simpleType* contains the other constructs, such that increasing the usage of *simpleType* will also increase the usage of those constructs. According to Table 1, the usage of *simpleType* has grown around 20% and its associated constructs have grown even more. Despite the general usage inscrease of most constructs, surprinsingly, the use of *key* and *keyref* still remains low (less than 10%). This might be an indication that, in most XML applications, user-defined keys are still the usual design choice.

Finally, the construct *notation* specifies the format of non-XML data. It was the least used one in 2005 and it does not appear in any of the documents in 2008. As already pointed out by van der Vlist [17], notations were very rare in real world applications. Now, we have just shown, empirically, that notations are not used at all.

Internal Usage

Here, we discuss some of the most interesting results presented on the last two columns of Table 1. Half of the constructs had their presence within XSD files decreased (while the other half increased). One of the possible reasons is, again, the users becoming more familiar with different constructs. One possible scenario is more people using different constructs with low internal usage, which is common in a learning phase. Since we calculate the average, the low usage in new files may have caused the overall decreasing. Another explanation, still related to learning, is that some

Table 3: Usage of constructs all, choice, sequence and group in complex content model

Construct	Internal Usage	
	2005(%)	2008(%)
All	8.61	6.77
Choice	11.52	14.43
Sequence	79.35	75.77
Group	0.51	3.03

constructs were replaced by others to better attend the applications requirements. For example, the lower usage of *enumeration* (which defines a list of possible values) may have been influenced by the higher usage of *pattern* (which defines regular expressions). Note that defining a list of possible values seems to be easier than defining a pattern. However, the latter is more powerful and requires more time to be mastered, while the former can be quickly learned.

Other Considerations

The results for general and internal usages may change when we analyze schemas of particular areas. For example, we can expect more use of *fractionDigits* in commercial and scientific applications. Hence, our explanations for the results must not be taken as true for any type of file collection.

Moreover, general and internal usage represent different aspects of a schema collection and do not have any direct relation. They also follow the 20/80 rule. Specifically, on the general usage columns, we can see that only 20% of the constructs appear in more than 80% of the XSD files. Likewise, on the internal usage columns, we find that 20% of the constructs represent 80% of all usage. It shows that usage of XML Schema constructs has been highly concentrated.

4.2 Complex Constructs

As we can see from Table 1, *complextType* is the second most used construct. It allows to create more sophisticated structures, to expand basic types, to provide more flexibility, and so on. Due to its importance and intricate structure, we provide a detailed analysis of its components. A closer view of other construct is left for future work.

Complex types are formed by either the simple or the complex content model. Simple content defines element attributes, whereas complex content describes the markup structure. Table 2 presents the distribution of simple and complex contents (internal usage). This table shows that the distribution has almost not changed along the last three years. This table also shows that a major part (more than 86%) of the *complexType* models in the schema contain complex contents. Therefore, we detail how those complex contents are actually employed next.

Group, All, Choice, and Sequence

In order to create complex content, the following constructs are available: *all*, *choice*, *sequence*, and *group*. Table 3 presents the usage of those constructs. The results on that table show a small variation from 2005 to 2008. The con-

Table 4: Usage of compositors with only one child-element

Construct	Internal Usage		General Usage	
	2005 (%)	2008 (%)	2005 (%)	2008 (%)
All	14.05	12.02	4.02	3.59
Choice	17.94	23.26	11.05	13.45
Sequence	42.75	48.08	77.39	77.58

Table 5: Nesting in composition constructors

Construct	Nesting	
	2005(%)	2008 (%)
Choice	8.86	5.46
Sequence	9.62	7.36

struct *sequence* (ordered elements) is still the most used, followed by *choice*, *all*, and *group*. The increase in using *group* may explain the decrease in using *all* and *sequence*. It is important to notice that *group* is defined outside *complexContent*, while the other three constructs can be embedded within it, which then may justify those numbers. We can also suppose that *all* and *sequence* were more used in *group* than in *choice*, since Table 3 shows that the usage of all those constructs has increased, but *choice*. Considering only Table 3, we could infer wrongly that *sequence* and *all* were being replaced by *choice*.

Table 4 presents the usage of compositors (*all*, *choice*, and *sequence*) with only *one* child-element. Those constructs define lists of elements. Hence, we could expect that a list would have more than one element. However, in our dataset, the reality is a little bit different. In 2005, it was common to include an only child-element in those compositors. In 2008, we confirmed such an interesting tendency. In the *sequence* construct we can observe that the usage of an only child-element is highly common (48% of internal usage). Perhaps the proportion observed in here can be a consequence of the usage found in Table 3. In many cases, we can use any of those constructs to build the same structure with one child-element. However, as we usually apply *sequence*, we tend to use it even when it is not the only option.

Nesting

Another interesting information is the presence of nested constructions within complex types. The usage of nesting shows how complex the schema structure is. Table 5 presents usage of nesting among compositors, except *all* which cannot be used as a particle [17]. According to Table 5, nesting is not really used in practice and its usage has decreased. Table 6 shows the usage of nesting in XSD files (at least one compositor using nesting). We can observe that most of the files do not use nesting among compositors as well.

We believe that the reasons for results from Table 5 and 6 are twofold. First, real world applications do not demand complex nestings. Second, users are not prepared enough to make an elaborate use of those kinds of constructs.

Table 6: General Usage of nesting in composition constructors

Type	General Usage	
	2005 (%)	2008 (%)
Nesting	32.66	33.18
No nesting	67.34	66.82

Table 7: General Usage of extension and restriction in simple and complex types

Derivation	Simple Type		Complex Type	
	2004 (%)	2008 (%)	2004(%)	2008 (%)
extension	27.00	37.84	37.00	34.53
restriction	73.00	82.88	7.00	7.62

Derivation: Extension and Restriction

In 2004, a study presented in [6] compared the features from DTDs and XSDs, with focus on expressive power. One important feature (from the point-of-view of language power) is the derivation of new types. Both simple and complex types may be derived by *extension* and *restriction*. In summary, there are four possibilities: (*i*) we can derive a complex type from a simple type by extension, and then add attributes to elements; (*ii*) we can extend a complex type and then add sequence of elements to its content model or add attributes; (*iii*) we can restrict a simple type and limit its acceptable range of values; and (iv) we can restrict a complex type and limit its acceptable range of subtrees.

That study considered 93 XSDs (collected in 2004) and it counted within how many files those four types of derivation were defined. We measured those as well in 2008, and Table 7 presents both results (from their study in 2004 and from ours in 2008). Note that there is an increase in the derivation of simple types, while there is practically no change in the derivation of complex types.

5. RELATED WORK

In this section, we present some related work divided in two parts. First, we overview some research papers on XML data management that depend on the information stored on the schemas. Second, we discuss other publications that evaluated actual schemas in use.

5.1 XML Data Management and Schemas

The management of XML data by a native or an XML-enabled DBMS has been widely discussed. Two central questions are how to store and how to query the data (for example [2, 4, 7, 9, 11, 15, 18], just to cite a few). We can divide those approaches into two categories: (*i*) those that do not depend on schema definition, such as [9] and [11]; and (*ii*) those that depend on schema definitions. The second category can be further divided in those that employ DTDs [4, 15] and XSDs [2, 7, 18].

Our empirical evaluation can be of major value to research and industry work similar to those aforementioned, as well as to those on information integration [5], schema evolu-

tion [13], and web services [19]. We offered a snapshot, an X-Ray, on what features are currently most and least used in XSDs. Therefore, one can decide to invest more time on optimizing the most frequently used constructs, leaving the least used ones for a second moment. Furthermore, we also show a trend in using more sophisticated constructs. Such a trend can have a positive impact on design decisions of new applications as well.

5.2 Actual Schemas in Use

An early study discussed how real DTDs were like [10]. Such a study evaluated files from a DTD repository and presented different statistics considering local and global properties, such as syntactic complexity, ambiguity, determinism, reachability, recursions, path sizes, among others. It is very similar to our study in spirit, since it wanted to provide an overview of currently in-use DTDs. However, ours deals with a more complex, powerful schema definition language.

A later study compared the features from DTDs and XSDs [6]. It considered DTDs and XSDs files from the Web and focused on finding out which features from XML Schema, that are not allowed in DTDs, are more used in practice and how sophisticated the features employed (in both languages) are in practice. The main conclusion is that the expressive power from real world XSDs are mostly equivalent to that of DTDs. However, note that its focus is on expressive power, while ours is on the actual schema structure. Hence, our study is complimentary to that one.

Also related to our work is [3], which presents a study over features of XML *documents* (schema instances) from the Web. It presented statistics on document distribution, schema usage, document internal features, among others. Specifically, the internal features are broken down on node distribution, size, depth, element and attribute fan-out, and recursion. That study focuses on the actual XML documents, while our focuses on the schema definition using XSD. Nevertheless, our research work complements all those aforementioned by providing an X-Ray on XSDs files.

6. CONCLUDING REMARKS

In this paper, we provided an X-Ray on the structural features of XSD files available on the Web. We considered files from the Web mainly because they are not specialized in a limited set of areas (e.g., bio or math data) such as those files found in some XSD repositories. Our evaluation showed the most and the least frequently used XSD constructs. We also presented an evolution on the usage of those constructs, by considering XSD files available on the Web in March 2005 and in November 2008. From such an evolution, we would like to emphasize three findings: (*i*) there is a general increase in the use of all constructs; (*ii*) the construct *notation* has disappeared from XSDs files; and (*iii*) the 20/80 rule, where 20% of the constructs represent more than 80% of all internal usage, and 20% of the constructs appear in more than 80% of all XSD files. Finally, we did not exhaustively interpret our results, leaving them open for further discussion as well. Nevertheless, our empirical evaluation provides relevant information for developers of XML applications, tools, and algorithms in which the schema has a distinguished role.

7. REFERENCES

[1] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 1999.

[2] S. Amer-Yahia, F. Du, and J. Freire. A Comprehensive Solution to the XML-to-Relational Mapping Problem. In *Procs. of WIDM*, pages 31–38, 2004.

[3] D. Barbosa, L. Mignet, and P. Veltri. Studying the XML Web: Gathering Statistics from an XML Sample. *World Wide Web Journal*, 8(4):413–438, 2005.

[4] M. Benedikt, W. Fan, and F. Geerts. XPath Satisfiability in the Presence of DTDs. In *Procs. of PODS*, pages 25–36, 2005.

[5] P. A. Bernstein and L. M. Haas. Information Integration in the Enterprise. *Commun. ACM*, 51(9):72–79, 2008.

[6] G. J. Bex, F. Neven, and J. V. den Bussche. DTDs versus XML Schema: A Practical Study. In *Procs. of WebDB*, pages 79–84, 2004.

[7] P. Bohannon et al. From XML Schema to Relations: A Cost-Based Approach to XML Storage. In *Procs. of ICDE*, pages 64–75, 2002.

[8] T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. M. F. Yergeau. *Extensible Markup Language (XML) 1.0*. W3C, 5th edition, November 2008. http://www.w3.org/TR/REC-xml/.

[9] D. Che, K. Aberer, and T. Özsu. Query Optimization in XML Structured-Document Databases. *The VLDB Journal*, 15(3):263–289, 2006.

[10] B. Choi. What are real DTDs like? In *Procs. of WebDB*, pages 43–48, 2002.

[11] D. Florescu and D. Kossmann. Storing and Querying XML Data using an RDMBS. *IEEE Data Eng. Bull.*, 22(3):27–34, 1999.

[12] D. Lee and W. W. Chu. Comparative Analysis of Six XML Schema Languages. *SIGMOD Record*, 29(3):76–87, 2000.

[13] M. M. Moro, S. Malaika, and L. Lim. Preserving XML Queries during Schema Evolution. In *Procs. of WWW*, pages 1341–1342, 2007.

[14] S. Nestorov, S. Abiteboul, and R. Motwani. Infering Structure in Semistructured Data. *SIGMOD Record*, 26(4):39–43, 1997.

[15] J. Shanmugasundaram et al. Relational Databases for Querying XML Documents: Limitations and Opportunities. In *Procs. of VLDB*, pages 302–314, 1999.

[16] H. S. Thompson et al. *XML Schema Part 1: Structures*. W3C, 2nd edition, October 2004. http://www.w3.org/TR/xmlschema-1/.

[17] E. van der Vlist. *XML Schema*. O'Reilly and Associates, 1st edition, June 2002.

[18] I. Varlamis and M. Vazirgiannis. Bridging XML-schema and relational databases. A system for generating and manipulating relational databases using valid XML documents. In *Procs. of ACM DocEng*, pages 105–114, 2001.

[19] Q. Yu et al. Deploying and Managing Web Services: Issues, Solutions, and Directions. *The VLDB Journal*, 17(3):537–572, 2008.

Rethinking Cost and Performance of Database Systems

Daniela Florescu
Oracle Corp.
dana.florescu@oracle.com

Donald Kossmann
28msec Inc. & ETH Zurich
donald.kossmann@28msec.com

ABSTRACT

Traditionally, database systems were optimized in the following way: "Given a set of machines, try to minimize the response time of each request." This paper argues that today, users would like a database system to optimize the opposite question: "Given a response time goal for each request, try to minimize the number of machines (i.e., cost in $)." Furthermore, this paper gives an example that demonstrates that the new optimization problem may result in a totally different system architecture.

1. INTRODUCTION

If you do DB research, then you optimize something. The fun of research is to define your own *something*; that is, define the constraints and the metrics of your optimization problem. If we are honest, however, the database community has pretty much answered and optimized the same question: "How to make a DBMS faster?" The only fun was defining the "scenario" in which to make the DBMS faster. Possible scenarios were new kinds of queries (e.g., analytics), new data types (e.g., images, polygons, or XML), and new architectures (e.g., streaming data). Nevertheless, the y-axes of the graphs of most SIGMOD and VLDB papers are the same and labelled with "[secs]" response time or "[tps]" throughput. Furthermore, the constraints such as strong consistency (i.e., ACID transactions) and the available hardware resources were fixed.

The purpose of this paper is to re-think what *better* means. That is, this paper tries to define how the y-axes of the graphs of future SIGMOD and VLDB papers should be labelled. The paper argues that those labels should change in the future. In a nutshell, rather than using "[secs]" or "[tps]", the y-axes should be labelled with "$". Furthermore, *consistency* should be a parameter that is varied when experimenting with a system. In other words, the purpose of this paper is to redefine the database optimization problem: define new constraints and different metrics to optimize.

This paper is motivated by several trends. Obviously, the most important trend are changing requirements for many applications. Many applications on the Web simply do not require strong consistency as mandated by the ACID paradigm; instead, these applications require scalability to millions of users and they require that no user is ever blocked by any other user. Furthermore, these applications require that all users get an answer within a second: There is no batch processing on the Internet and Web users do not understand why processing a complex query might take longer than processing a simple query. Another trend is the growing complexity of software systems and the distributed, service-oriented architecture of most software systems. Furthermore, there are technology trends such as cloud computing and large data centers with cheap hardware that have changed the assumptions of modern software architectures. The first contribution of this paper is to detail how these trends have changed the DB optimization problem.

A second contribution of this paper is a discussion on how the *new* database optimization problem might impact the architecture of a modern database system and possibly change some of the fixed parameters of traditional database research. As a starting point, this paper describes the architecture of the 28msec application server, an integrated database system, Web server, and virtual machine that runs on the Amazon Web Services platform (AWS).

The remainder of this paper is structured as follows: Section 2 lists the *new* requirements for modern database systems. Section 3 presents the architecture of the 28msec application server as an example for how these new requirements can be met. Section 4 discusses related work. Section 5 contains conclusions.

2. WHAT DO USERS CARE ABOUT?

Essentially, the requirements of users have not changed: They want it *all*. Also, the definition of *all* has not changed much: zero cost, zero response time, infinite throughput, infinite scalability in terms of users supported, linear scalability with the number of machines added, 100 percent predictability of cost and performance, ACID-style transactions, 100 percent availability for read and write requests, and flexibility to customize the system towards individual needs at any point in time with little effort. What has changed are the priorities and the constraints if the users cannot have it all. In a nutshell, the *traditional* database optimization problem can be defined as follows:

> *Given a set of hardware resources and guaranteeing full data consistency (i.e., ACID transactions), minimize the response time of requests and maximize the throughput of requests.*

The *new* database optimization problem can be defined as follows:

Feature	*Trad. DB*	*New DB*
Cost [$]	fixed	minimize
Performance [secs and tps]	optimize	fixed
Scalability [machines]	maximize	fixed
Predictability [$ and secs]	-	fixed
Consistency [%]	fixed	maximize
Flexibility [#variants]	-	maximize

Table 1: Traditional vs. New DB Optimization Problem

> *Given performance requirements of an application (peak throughput; maximum tolerable response times), minimize the required hardware resources and maximize the data consistency.*

Those two problems are not in conflict and, clearly, many techniques to optimize traditional database systems are applicable in the new world. As shown in Sections 3 and 4, however, the subtle differences may have significant impact on the architecture of a system. Table 1 summarizes these differences in the problem formulation; these differences are described in the remainder of this section in a bit more detail.

Cost. Again, this paper argues that *cost* as measured in $ is the main metric that needs to be optimized. The big question is no longer how fast a database workload can be executed or whether a particular throughput can be achieved; instead, the question is how many machines are necessary to meet the performance requirements of a particular workload. Cloud computing, increased cost for power and cooling, and the popularity of services like Amazon Web Services (AWS) have significantly contributed to this observation. Furthermore, hardware resources are no longer a one-time investment; instead, hardware resources are a significant lineitem on the monthly IT bill. Furthermore, cloud computing and AWS have made this cost a *continuous* metric (rather than *discrete* metric): The more hardware you consume, the more you pay and consumption is measured in a fine-grained way as CPU cycles and bytes of storage needed. Using AWS, costs are metered in *millidollars*. Traditionally, hardware resources have been provisioned in the granularity of machines; as a result, incremental costs are in the order of *kilodollars*.

The need to put cost into the performance metric has long been realized as part of the TPC benchmarks [1]. It has also been endorsed as part of the Mariposa project [12]. Unfortunately, most database research today still ignores this metric and even the TPC benchmarks mix cost and performance into a tps/$ metric. This paper argues for a more extreme approach to make cost alone *the* metric for most experiments. Furthermore, cloud computing services like AWS have made this metric comparable. Everybody can run experiments with various algorithms on AWS and report on "Amazon $" consumed. In the TPC benchmark reports, the $ metric is often questionable and controversial.

Performance (Response Time and Throughput). In most businesses, performance is a *constraint* and not an optimization goal. Google, for instance, is fast enough as it is; improving the response time of Google would not help. There are only few applications for which "faster is better" without any limits: Algorithmic trading in the financial sector is one of these rare examples. In all interactive applications (e.g., Web applications), a response time of a few hundred milliseconds is good enough in order to make users happy. In terms of througput, the requirement is to sustain a particular peak workload. Again, the big question in most IT scenarios is no longer whether it is *possible* to sustain the load: (Almost) everything is possible. Again, the question is at what *cost* it can be done; the "cheaper the better."

Another reason to put less emphasis on performance of a DBMS is that in practice, many performance problems are no longer caused by the DBMS. Modern IT systems are very complex and the DBMS is only one component of many. Again, improving the performance of the DBMS might not help.

Scalability. One basic assumption made today and in the discussion of the last two paragraphs on cost and performance is that every IT problem can be solved by throwing hardware (i.e., money) at it. Therefore, infinite scalability is a *must* in many modern IT systems. Ultimately, every business wants to grow, even if it is small at the beginning: Scalability involves that the IT costs grow linearly with the business and that this growth is unlimited. Unfortunately, this requirement is not met by many traditional database systems for which the cost function is a step function and the scalability is bound.

Predictability. For many businesses, predictability is a *must* in the same way as scalability. In other words, optimizing for the 99 percentile has become more imortant than optimizing for the average or mean. Here, predictability refers to both the predictability of performance and cost. Most database vendors have made a great deal of effort in order to reduce the cost of administration and make the performance of their systems more robust; nevertheless, database administration and provisioning of hardware resources for large-scale database applications is still a black art.

Consistency. ACID is great! SOA is greater!

With ACID, developers need not worry about consistency and can focus on the application logic. SOA (service-oriented architecture) helps developers to structure and more importantly evolve their applications.

Unfortunately, ACID and SOA are like water and oil: they do not mix well. ACID requires full control of all data management activities carried out by a transaction, whereas SOA mandates autonomy of all services involved in a transaction. While standards such as XA have helped to support distributed ACID-style transactions in a service-oriented infrastructure, supporting ACID transactions in large-scale distributed systems still remains painful. Fortunately, ACID transactions and strong consistency are rarely needed, as observed in a recent keynote by W. Vogels [16] and a paper by P. Helland [8]. Even for mission critical operations, a lower-level of consistency is often sufficient. [3] argues that the level of consistency should be flexible and that there should be a trade-off between the cost and consistency of the data. Furthermore, both [16] and [3] argue for a different definition of consistency levels along the lines of [15], as used for the design of large-scale distributed systems, rather than SQL isolation levels, as implemented by today's generation of commercial database systems. Again, this perspective is mandated by the importance of SOA as a design principle.

The requirement to provide 100 percent read and write availability for all users has also overshadowed the importance of the ACID paradigm as the gold standard for data consistency. In large Web

applications (e.g., Amazon, eBay, expedia, etc.), no user is ever allowed to be blocked; in particular, no user is allowed to be blocked by the actions of another user. Again, this requirement takes priority over the goal to achieve strong consistency. As a result, it is better to design a system so that it deals with and helps resolve inconsistencies, rather than having a system that prevents inconsistencies under all circumstances. "Shit happens" even with ACID and disregarding this observation limits the capabilities of a system. As a result, consistency is an optimization goal in modern IT systems in order to minimize the cost of resolving inconsistencies and not a constraint as in traditional database systems.

Flexibility. Flexibility is the ability to customize a software system to the individual needs of a customer. Flexibility makes it also easier, faster and cheaper to make new releases of a system. Flexibility is, thus, measured in the number of variants of a system that are deployed. Flexibility is particularly important for applications that have a diverse user base; typical examples are enterprise applications such as SAP R/3 or Oracle Finance. As applications become more complex, flexibility as a design goal has become increasingly important. A recent indicator for the importance of this requirement is the success of the Salesforce AppExchange technology. Traditional OLTP systems such as the credit/debit systems of the 1970s did not have this requirement so that flexibilty has not been a pressing optimization goal for traditional database management systems. Obviously, providing good performance and scalability is much easier if the system need not be flexible.

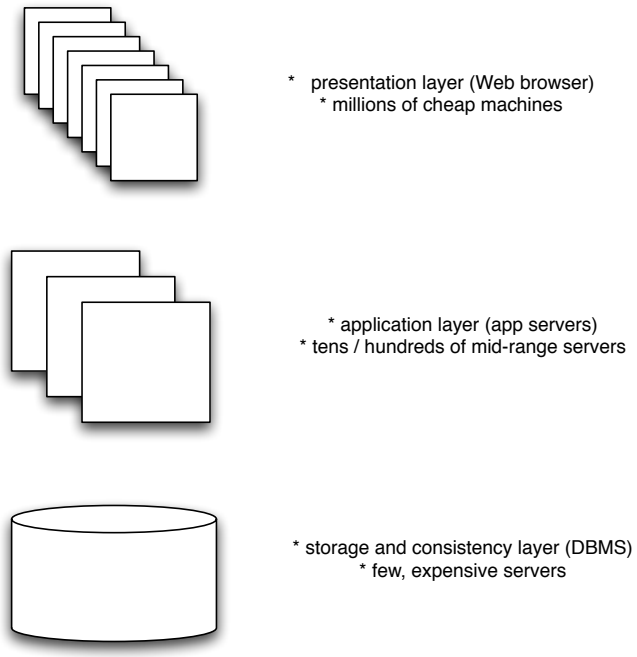


Figure 1: Classic Database Architecture

3. DOES IT MATTER?

This section revisits the *classic* database architecture that was designed to optimize the *traditional* DB problem (Table 1). Furthermore, this section describes a new, different database architecture that was designed to optimize the *new* DB problem.

3.1 Traditional Databases Revisited

Figure 1 shows the *classic* three-tier architecture in order to build database applications. This architecture was pioneered by SAP and variants of this architecture are still the state-of-the-art in order to build database applications. All requests are initiated by users at the presentation layer, e.g., using a Web browser. The application logic is encoded in the middle tier; the middle tier also involves functionality such as the Web server. All data management is carried out at the lowest tier using a DBMS (e.g., IBMD DB2, Microsoft SQL Server, MySQL, Oracle etc.).

The beauty of this architecture is that it serves extremely well the requirements of traditional database systems (Table 1). The main constraint of keeping data consistent is achieved in the bottom tier, inside the database server. The main optimization goal of minimizing response times and maximizing throughput is achieved by a series of techniques that have been developed over the last four decades in all tiers: caching, indexing and data partitioning, to name just a few. Furthermore, scalability is achieved on the two top tiers: at those tiers, the architecture of Figure 1 scales almost infinitely. Scalability, however, is limited at the bottom tier.

Unfortunately, the three-tier architecture of Figure 1 is not a good fit for the *new* requirements of Table 1. Most importantly, this architecture is expensive. Typically, significant investments are needed for the bottom tier, thereby involving expensive hardware (rather than cheap hardware). The hardware must be provisioned for *expected peak performance*, which can be orders of magnitudes higher than the *real average performance*. As a result, a great deal of hardware is wasted and it is difficult to re-use those unused hardware resources for other purposes. Furthermore, maintaining the DB infrastructure is expensive and does not get automatically cheaper, like anything else. Finally, the DB software itself can be a cost factor: Most of the functionality is not needed by an application so that clients are forced to pay for unneeded features. This situation is improving with open source database systems such as PostGres and MySQL; nevertheless, many organizations are still locked into expensive commercial database systems.

The classic three-tier DB architecture and the way it is implemented today also does not meet the predictability requirements. With many concurrent clients accessing the same database, it is virtually impossible to understand what is happening at the database level. Also, today's generation of database products have just not been designed for predictability. Furthermore, as mentioned above, scalability is limited in this architecture to the top two tiers.

Flexibility is another design goal that is not supported well by traditional database architectures. The reason is that, in the state-of-the-art, different technologies are used at all three tiers. SQL and the relational model are used at the bottom tier; OO is used in the middle tier; and XML/HTML with some client-side scripting are used at the top tier. As a result, a change in functionality (i.e., customization) must be implemented at all three tiers, thereby using three different technologies. Such customizations are expensive to implement and difficult to test.

It is interesting to re-iterate two design principles of the classic three-tier architecture for database applications. The first principle is *control*. At the bottom tier, the DBMS controls all hardware resources and all accesses to the data. The second design principle is typically referred to as *query shipping* [6]. Query shipping means that as much functionality as possible is pushed to the bot-

tom tier; e.g., as stored procedures or exploiting other features of modern database systems (e.g., new data types and query capabilities). On the positive side, both of these design principles have helped to achieve the two most important design goals of traditional database systems: consistency and performance. Query shipping was also motivated by the business model of most DBMS vendors: New DBMS licenses could only be sold by providing more functionality. On the negative side, these two design principles have turned DBMSes into monolithic monsters, getting bigger and bigger, never smaller. This trend has hurt predictability, flexibility, and scalability. Furthermore, these two design principles have made IT systems expensive because expensive hardware is needed in order to sustain the load at the bottom tier and because the complexity of modern DBMS has its price. Not surprisingly, therefore, the architecture presented in the next subsection has exactly the inverse design principles.

3.2 A New Architecture

Figure 2 shows an architecture designed for the *new* database optimization problem. Comparing Figures 2 and 1, both architectures look similar at first glance, but the differences are significant. The key ideas of the architecture of Figure 2 can be summarized as follows:

- Consistency is not handled at the storage layer, but at the application layer. At the bottom layer, there is only a large-scale distributed storage such as the one provided by Amazon S3.
- Consistency in the middle-tier is achieved by distributed protocols such as those devised in [15]. That is, there is no entity that controls all accesses to the storage. Instead, all application servers that access data agree on *conventions* on how to read and update data stored in the bottom layer.
- Cheap hardware can be used in *all* layers. All layers are designed to scale to thousands, if not millions of machines. The whole architecture is designed such that any node can fail at any time. At the bottom-tier, fault tolerance is achieved by replicating data and by giving poor consistency guarantees only (i.e., eventual consistency). In the upper layers, machines are stateless so that the worst that can happen if a machine fails is that work of the currently active transactions is lost.
- Essentially, the traditional DBMS has disappeared from the architecture of Figure 2. The main DBMS functionality, i.e., transaction and query processing, has been integrated into the application layer.
- It is possible to run both the application layer and the presentation layer on an end user's machine. Again, processes at both layers are stateless and the architecture has been designed so that processes at both of these layers can fail at any time.

This architecture has been implemented by 28msec as part of its Sausalito product[1]. Sausalito integrates a virtual machine, data management system, data stream processor, queue, and Web server in a single portable platform. This sounds like yet another monster, even more than a DBMS, but in fact it is not. At the moment, the footprint of the whole platform is about 140 mega-bytes, mostly because it has a leaner and more focused functionality.

[1]More information can be found at http://www.28msec.com.

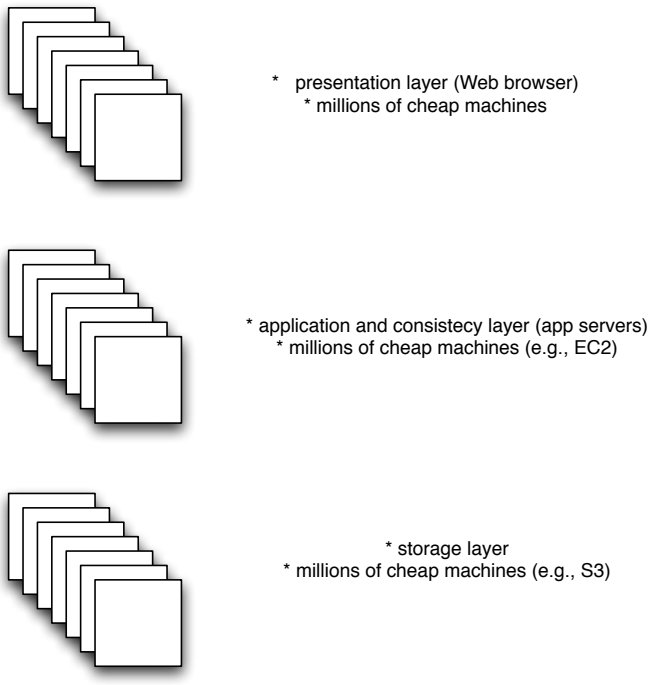


Figure 2: 28msec Architecture

With Sausalito, all data and (pre-compiled) application code is stored as blobs using Amazon S3. Each HTTP request (e.g., a click of a user in a Web browser or a Web Service call from an application) is processed in the following way: Using Amazon EC2 and its scheduling and load-balancing service, an available EC2 server is selected to process that request. This EC2 server loads from S3 the pre-compiled code which handles this request. This pre-compiled code is then interpreted on the EC2 server by the Sausalito runtime system, thereby possibly involving accesses to objects in the database which are also stored as blobs in S3. All EC2 servers are stateless and can fail at any time. If the load increases (decreases), EC2 servers can be added (dropped, respectively). Amazon S3 uses cheap hardware at the storage layer and achieves availabilty by replicating all blobs multiple times across different data centers. Synchronoziation of concurrent accesses to the same data (from multiple EC2 servers) is effected using the protocols described in [3].

As a programming language to implement all application logic and database access, Sausalito supports XQuery (including updates and scripting extensions). Sausalito uses XQuery because it supports all Web standards nicely (in particular, REST and Web services), integrates database queries, and is sufficient to build full-fledged Web-based applications. For the same purpose, Google's AppEngine uses Python with an embedded SQL-like dialect for database access. Microsoft relies on the .NET programming languages with LINQ.

Obviously, there is little hope that any system (including Sausalito) that implements the architecture of Figure 2 is able to match

the state-of-the-art in terms of performance and consistency. With regard to consistency, it is not possible to implement strong consistency, high availability and scalability at the same time, according to Brewer's CAP theorem [7]. With regard to performance, state-of-the-art DBMSes have been effectively optimized for fourty years. The architecture of Figure 2, however, works well for the other features of Table 1. As shown with Sausalito, it can be implemented in a cost-effective way by using cheap hardware everywhere. Furthermore, it was designed for scalability. Flexibility is achieved by a simplified platform and by using only one programming and data model, rather than different models at all levels. As shown in [3] for many workloads, predictable cost and performance can be achieved, too.

From the discussion of this section, it should have become clear that the architecture of Figure 2 was devised with the opposite design principles as compared to the architecture of Figure 1. Rather than controlling all reads and writes to the data, there are conventions that rule access to data in a distributed and loosely coupled way; these conventions are implemented as distributed protocols [15, 3]. Instead of moving functionality to the data, the storage layer has a minimal "get" and "put" interface. All other functionality is implemented at the application layer. One particular aspect is security: Security must be implemented at the application level, thereby encrypting all data stored in the storage layer and controlling the dissemination of keys to access that data. We expect that implementing security is not more difficult in the architecture of Figure 2 than in the architecture of Figure 1 because security must be considered at the application and presentation layers anyway.

4. RELATED WORK

Obviously, we are not the first to question modern DBMS architectures. All ideas presented in this paper have been floating around for a while and the purpose of this position paper is to try to put the pieces of the big picture together. Obviously, the recent hype about cloud computing and Amazon Web Services (e.g., S3 and EC2) and the Google AppEngine services have cleared the path towards re-thinking the cost of modern information systems. The impact of SOA design principles on database application architectures has been addressed recently in talks by Vogels [16] and Helland [8]. Furthermore, there have been a number of relevant techniques that take a different spin towards database optimization such as Eddies [2] and query optimization for the expected and worst case [4].

In the late 1990s, people realized that big application systems such as SAP use a DBMS only as a glorified file system with a built-in persistent B-tree [5]. At the time, the reaction of the DBMS vendors was to ask SAP how they could *extend* their products in order to better meet SAP's requirements. The right question to ask would have been how to repackage DBMS functionality in order to better integrate into the application architecture. Unfortunately, this question was never asked.

The most relevant recent work on database architecture is Stonebraker et al.'s observation that "one size does not fit all" [14, 13]. That work showed the current short-comings of state-of-the-art database management systems in several important application areas: stream processing, decision support (i.e., OLAP), and transaction processing (i.e., OLTP). Comparing that work with the observations presented in this paper, there is a great deal of commonality. First, both lines of work agree that state-of-the-art database systems do not seem to be optimal for anything anymore. Second and more importantly, both lines of work argue that DB functionality can be *repackaged* effectively and in a lean way in order to better match the needs of applications. Third, while Stonebraker et al.'s work is focussed on performance and achieving orders of magnitude performance improvements, the same arguments can actually be applied to *cost* which is the focus of our work. Fourth, the architectural principles discussed in Section 3.2 do not contradict in any way with the technology proposed in [14, 13]. In some sense, our work builds on top of the results of Stonebraker et al. and is not at all in conflict with that work; Stonebraker et al.'s work is geared towards how to build modern data management systems (e.g., column store, on-the-fly data processing, and simplified synchronization) whereas the purpose of this paper is to present the bigger picture and how to integrate that technology into the whole application and technology stack.

There has also been a great deal of work on improving the scalability of database management for the Web. Most significantly, there has been work on caching and materialized Web views; e.g., [10, 17, 11]. The goal of all that work is to increase the scalability of modern Web-based systems by off-loading the database. Furthermore, as mentioned earlier, cost (measured in $) has played a role in the TPC benchmarks. It has also been used in a number of frameworks in order to improve performance and quality of service in a distributed information system. Prominent examples are the Mariposa system [12] and the "Quality Contracts (QC)" framework [9].

5. CONCLUSION

The purpose of a database system is to help developers to write applications, deploy their applications and run their applications. In the 1960s, most of the database applications were simple debit/credit transactions: database systems were a huge help and almost solved the whole problem: It was well affordable to spend a large portion of the IT budget on database software and administration. In the meantime, applications have grown and databases (and middleware) only solve a relatively small fraction of the problem. As a result, database systems are creating often more pain than they actually help because database systems are just one of many components, yet highly demanding and often dictating the whole application architecture. In a nutshell, the assumptions and the goals of the usage of database systems have changed over the last fourty years, when the first generation of database systems was built.

The purpose of this paper is to rethink the assumptions and goals and, thus, redefine the *DB problem*. The paper tried to do that for Web-based applications such as, e.g., an online bookstore or a car-pooling service. The result was a new problem statement, and not surprisingly a new architecture, and a different packaging of database functionality. Obviously, these results are biased by the particular application scenarios of Web-based, interactive applications. Other applications such as data warehousing and decision support may result in different results. Furthermore, the proposed new database application architecture is not mature: While it seems that big players like Google and start-ups like 28msec are moving to such an architecture, there is still a great deal of work needed before there are products which are ready for prime time. Nevertheless, the key message of this paper stays the same; citing Goetz Graefe (in the mid 1990s): "Database vendors should not be building a Ferrari - they should be building a Ford Taurus."

Acknowledgments. We would like to thank Paul Hofmann (SAP) for discussions and many helpful comments on this paper.

6. REFERENCES

[1] Anon et al. A measure of transaction processing power. *Datamation*, 1984.

[2] R. Avnur and J. Hellerstein. Eddies: Continuously adaptive query processing. In *Proc. of ACM SIGMOD*, pages 261–272, Jun 2000.

[3] M. Brantner, D. Florescu, D. Graf, D. Kossmann, and T. Kraska. Building a database on S3. In *Proc. of ACM SIGMOD*, pages 251–264, Jun 2008.

[4] F. Chu, J. Halpern, and J. Gehrke. Least expected cost query optimization: What can we expect? In *Proc. of ACM PODS*, pages 293–302, Jun 2002.

[5] J. Doppelhammer, T. Höppler, A. Kemper, and D. Kossmann. Database performance in the real world: TPC-D and SAP R/3. In *Proc. of ACM SIGMOD*, pages 219–230, Jun 1997.

[6] M. Franklin, B. Jonsson, and D. Kossmann. Performance tradeoffs for client-server query processing. In *Proc. of ACM SIGMOD*, pages 149–160, Jun 1996.

[7] S. Gilbert and N. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant Web services. *SIGACT News*, 33(2):51–59, 2002.

[8] P. Helland. Life beyond distributed transactions: An apostate's opinion. In *Proc. of CIDR Conf.*, pages 132–141, Jan 2007.

[9] A. Labrinidis, H. Qu, and J. Xu. Quality contracts for real-time enterpises. In *Business Intelligence for the Real-Time Enterprises*, pages 143–156, August 2007.

[10] A. Labrinidis and N. Roussoupoulos. Webview materialization. In *Proc. of ACM SIGMOD*, pages 367–378, Jun 2000.

[11] Q. Luo, S. Krshnamurthy, C. Mohan, H. Pirahesh, H. Woo, B. Lindsay, and J. Naughton. Middle-tier database caching for e-business. In *Proc. of ACM SIGMOD*, pages 600–611, Jun 2002.

[12] M. Stonebraker, P. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu. Mariposa: A wide-area distributed database system. *VLDB Journal*, 5(1):48–63, 1996.

[13] M. Stonebraker, C. Bear, U. Cetintemel, M. Cherniack, T. Ge, N. Hachem, S. Harizopoulos, J. Lifter, J. Rogers, and S. Zdonik. One size fits all? Part 2: Benchmarking studies. In *Proc. of CIDR Conf.*, pages 173–184, Jan 2007.

[14] M. Stonebraker, S. Madden, D. Abadi, S. Harizopoulos, N. Hachem, and P. Helland. The end of an architectural era (it's time for a complete rewrite). In *Proc. of VLDB Conf.*, pages 1150–1160, Sep 2007.

[15] A. Tanenbaum and M. van Steen. *Distributed Systems: Principles and Paradigms*. Prentice Hall, Upper Saddle River, NJ, 2002.

[16] W. Vogels. Data access patterns in the Amazon.com technology platform. In *Proc. of VLDB*, page 1, Sep 2007.

[17] K. Yagoub, D. Florescu, V. Issarny, and P. Valduriez. Caching strategies for data-intensive web sites. In *Proc. of VLDB*, pages 188–199, Sep 2000.

Logical Foundations of Relational Data Exchange

Pablo Barceló
Department of Computer Science, University of Chile
pbarcelo@dcc.uchile.cl

1 Introduction

Data exchange has been defined as the problem of taking data structured under a *source* schema and materializing an instance of a *target* schema that reflects as accurately as possible the source data [19]. In the last years, the need for data exchange applications has increased, particularly due to the proliferation of web data in various formats (relational, XML, RDF, etc) and the emergence of e-business applications that need to communicate data yet remain autonomous. Responding to this demand, commercial data exchange systems have been built recently [12].

Even though data exchange is an old and common data management problem, its most foundational aspects had not been studied until very recently. There are two reasons for this. First, most of the early research on databases concentrated on the stand-alone relational model, and much less on interoperability, integration, and exchange. Second, there were no solid foundation, nor even a proper formal model, for the problem of data exchange. Such a model was finally proposed in 2003 by Fagin, Kolaitis, Miller and Popa [19], and was quickly adopted as the right model for data exchange. A survey on the topic has already appeared in the premier conference on database theory, PODS, [30], and two workshops on exchange and integration of data have already been held [10, 39].

This survey is organized as follows. In Section 2, we present the basics of data exchange. One of the goals of data exchange is materializing a target instance that is consistent both with the source data and the specification of the relationship between the source and the target. Such a target instance is called a *solution* for the given source data. The work of Fagin et al. [19] identified a class of solutions, called *universal* solutions, with good properties for data exchange. We introduce the class of universal solutions in Section 3. In Section 4, we study the problem of the materialization of universal solutions. In particular, we show that there is a meaningful class of data exchange settings for which universal solutions are guaranteed to exist, if a solution exists at all, and, if that is the case, then a universal solution can always be constructed in polynomial time. Also in Section 4 we study the core of the universal solutions, which happens to be the smallest universal solution.

Database Principles Column. Column editor: Leonid Libkin, School of Informatics, University of Edinburgh, Edinburgh, EH8 9AB, UK. E-mail: libkin@inf.ed.ac.uk.

The notion of query answering in data exchange is presented in Section 5. After defining the semantics, which is based on the notion of certain answers (that is, those tuples that appear in answers for all solutions), we show that the problem of evaluating queries in data exchange becomes tractable for a relevant class of queries; namely, unions of conjunctive queries, which conform the logical core of the SQL query language. In Section 6 we show that other semantics can be meaningfully applied in data exchange. In particular, we present a semantics based on the class of universal solutions and a semantics based on the notions of closed-world assumption and incomplete information. Finally, we present in Section 7 a brief survey of other work in data exchange. Concluding remarks are in Section 8.

2 Data Exchange Settings

A *schema* $\mathbf{R}$ is a finite sequence $\langle R_1, \ldots, R_m \rangle$ of relation symbols, with each R_i having a fixed arity $n_i > 0$. An *instance* I *of* $\mathbf{R}$ assigns to each relation symbol R_i of $\mathbf{R}$ a finite n_i-ary relation $I(R_i)$. The *domain* of instance I is the set of all elements that occur in any of the relations $I(R_i)$. It is often convenient to define instances by simply listing the tuples attached to the corresponding relation symbols. Sometimes we use the notation $R(\bar{t}) \in I$ instead of $\bar{t} \in I(R)$, and call $R(\bar{t})$ a *fact* of I. Finally, a *dependency* over $\mathbf{R}$ is a sentence in some logical formalism, typically first-order logic (FO), with which we assume familiarity.

Given schemas $\mathbf{S} = \langle S_1, \ldots, S_m \rangle$ and $\mathbf{T} = \langle T_1, \ldots, T_n \rangle$, with no relation symbols in common, we denote by $\langle \mathbf{S}, \mathbf{T} \rangle$ the schema $\langle S_1, \ldots, S_m, T_1, \ldots, T_n \rangle$. Further, if I is an instance of $\mathbf{S}$ and J is an instance of $\mathbf{T}$, then (I, J) denotes an instance K of $\langle \mathbf{S}, \mathbf{T} \rangle$ such that $K(S_i) = I(S_i)$ and $K(T_j) = J(T_j)$, for each $i \in [1, m]$ and $j \in [1, n]$.

Definition 2.1 (Data exchange setting) *A* data exchange setting $\mathcal{M}$ *is a triple* $(\mathbf{S}, \mathbf{T}, \Sigma)$*, where* $\mathbf{S}$ *and* $\mathbf{T}$ *are disjoint schemas,* $\mathbf{S}$ *is called the* source *schema,* $\mathbf{T}$ *is called the* target *schema, and* Σ *is a finite set of dependencies over* $\langle \mathbf{S}, \mathbf{T} \rangle$.

Instances of $\mathbf{S}$ are called *source* instances, while instances of $\mathbf{T}$ are called *target* instances. It is usual in the data exchange literature to assume the existence of two disjoint and infinite set of values that populate instances. One is the set of *constants*, denoted by Const, and the other one

is the set of *nulls*, denoted by Var. The domain of a source instance is always contained in Const, while the domain of a target instance is contained in Const ∪ Var. We usually denote constants by lowercase letters $a, b, c, \ldots$, while nulls are denoted by symbols $\perp, \perp_1, \perp_2, \ldots$

In data exchange, the target instances that are consistent with both the source instance and the specification Σ are called solutions. Formally, given a source instance I, we say that the target instance J is a *solution for I under $\mathcal{M}$*, or simply a solution for I if $\mathcal{M}$ is clear from the context, if (I, J) satisfies every sentence in Σ.

Admitting the full expressive power of FO as a language for specifying dependencies in data exchange, easily yields to undecidability of some fundamental problems, like checking for the existence of solutions [18]. Thus, it is customary in the data exchange literature [19, 20, 30] to restrict the study to the class of settings $\mathcal{M}$, such that Σ can be split into two sets Σ_{st} and Σ_t that satisfy the following:

1. Σ_{st} consists of a set of *source-to-target* dependencies (stds), i.e. dependencies of the form $\forall \bar{x}\,(\varphi_{\mathbf{S}}(\bar{x}) \rightarrow \exists \bar{y} \psi_{\mathbf{T}}(\bar{x}, \bar{y}))$, where $\varphi_{\mathbf{S}}(\bar{x})$ and $\psi_{\mathbf{T}}(\bar{x}, \bar{y})$ are conjunctions of atomic formulas in $\mathbf{S}$ and $\mathbf{T}$, respectively; and
2. Σ_t is the set of *target* dependencies. It is the union of a set of *tuple-generating* dependencies (tgds), i.e. dependencies of the form $\forall \bar{x}\,(\varphi(\bar{x}) \rightarrow \exists \bar{y} \psi(\bar{x}, \bar{y}))$, where $\varphi(\bar{x})$ and $\psi(\bar{x}, \bar{y})$ are conjunctions of atomic formulas in $\mathbf{T}$, and a set of *equality-generating* dependencies (egds), i.e. dependencies of the form $\forall \bar{x}\,(\varphi(\bar{x}) \rightarrow x_i = x_j)$, where $\varphi(\bar{x})$ is a conjunction of atomic formulas in $\mathbf{T}$, and x_i, x_j are variables in $\bar{x}$.

From now on, and unless stated otherwise, we assume all data exchange settings to be of the form $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where $\Sigma = \Sigma_{st} \cup \Sigma_t$), for Σ_{st} a finite set of stds and Σ_t a finite set of tgds and egds. The intuition behind the different components of these settings is the following: Source-to-target dependencies in Σ_{st} are a tool for specifying which conditions on the source imply a condition on the target. But from a different point of view, one can also see them as a tool for specifying how source data gets translated into target data. In addition, the translated data must satisfy usual database constraints. This is represented by means of the target dependencies in Σ_t. It is important to notice that the data exchange settings described above are not restrictive from a database point of view. Indeed, tuple-generating dependencies together with equality generating dependencies precisely capture the class of *embedded* dependencies [17]. And the latter class contains all relevant dependencies that appear in relational databases, e.g. it contains functional and inclusion dependencies, among others.

Next example shows two interesting phenomena regarding solutions in data exchange. First, that solutions for a given source instance are not necessarily unique. Second, that there are source instances that have no solutions.

Example 2.2 *Consider a data exchange setting $\mathcal{M}$ in which $\mathbf{S}$ consists of the binary relations M and N, $\mathbf{T}$ consists of the ternary relation P and the binary relation Q, $\Sigma_t = \emptyset$ and Σ_{st} consists of the following stds (we implicitly assume universal quantification in front of all dependencies):*

$$\begin{aligned} M(x,y) &\rightarrow \exists w \exists z (P(x,y,z) \wedge Q(w,z)), \\ N(x,y) &\rightarrow \exists u P(x,y,u). \end{aligned}$$

Suppose we have a source instance $I = \{M(a,b), N(a,b)\}$. Since the stds in Σ_{st} do not completely specify the target, solutions for I are not unique up to isomorphism. For instance, one solution is:

$$J = \{P(a,b,\perp_1), P(a,b,\perp_2), Q(\perp_3, \perp_1)\},$$

where $\perp_1, \perp_2, \perp_3$ are values in Var *(nulls). Another solution, but with no nulls, is $J' = \{P(a,b,a), Q(b,a)\}$. Further, it is not hard to see that any other target instance that contains J or J' is a solution for I. Thus, I admits infinitely many solutions.*

Consider now the setting $\mathcal{M}'$ that extends $\mathcal{M}$ by adding the following egd to Σ_t: $P(x,y,z) \rightarrow x = y$. Then I has no solution under $\mathcal{M}$. Indeed, if there was at least one such solution J, then the first dependency in Σ_{st} implies that there is a fact of the form $P(a,b,z)$ in J, while the egd implies that that the constants a and b are equal, which is a contradiction. □

3 Universal Solutions

In this section we introduce a certain class of solutions that exhibit good properties for data exchange: the *universal* solutions. Notice, in Example 2.2, that the solution J' seems to be less general than J. This is because J' assumes that the values that witness the existentially quantified variables z and u, in the first and second std of Σ_{st}, respectively, are the same (namely, the constant a). It also assumes that the value that witnesses the existentially quantified variable w is the constant b. But none of these assumptions is part of the specification. On the other hand, solution J contains exactly what the specification requires. Since one of the basic problems in data exchange is materializing a target instance given a source instance, in this case one would like to materialize a solution like J rather than solution J'.

In order to give a precise mathematical definition of which solutions are the most general, we first have to define what a homomorphism between data exchange instances is. Let J and J' be two instances over the target schema $\mathbf{T}$ with values in Const ∪ Var. A *homomorphism* $h : J \rightarrow J'$ is a mapping from the domain of J into the domain of J', that is the identity on constants, and such that $\bar{t} = (t_1, \ldots, t_n) \in J(R)$ implies $h(\bar{t}) = (h(t_1), \ldots, h(t_n))$ is in $J'(R)$ for all $R \in \mathbf{T}$.

Definition 3.1 (Universal solutions) *Let J be a solution for I. Then J is a* universal *solution for I if for every solution J' for I, there exists a homomorphism $h : J \to J'$.*

Example 3.2 *Solution J' in Example 2.2 is not universal, since there is no homomorphism $h : J' \to J$. But it can be shown that J is a universal solution.* □

A universal solution is more general than an arbitrary solution because it can be homomorphically mapped into that solution. Furthermore, as we will see later, universal solutions possess good properties that justify materializing them (as opposed to arbitrary solutions). Unfortunately, universal solutions are not a general phenomenon. Indeed, [19] proved that there is a setting $\mathcal{M}$ and a source instance I, such that I has at least one solution under $\mathcal{M}$ but has no universal solutions. Thus, it is necessary to impose extra conditions on dependencies if one wants to make sure that the existence of solutions implies the existence of universal solutions. We study this issue in detail in the next section.

4 Materializing Solutions

One of the goals in data exchange is materializing a solution that reflects as accurately as possible the source data. Unfortunately, even the most basic problem of checking for the existence of solutions is undecidable:

Theorem 4.1 *[31] There exists a data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st} \cup \Sigma_t)$, such that the problem of deciding for a given source instance I, whether I has a solution under $\mathcal{M}$, is undecidable.*

Thus, one would like to restrict the class of dependencies allowed in data exchange settings, in such a way that it satisfies the following: (C1) the existence of solutions implies the existence of universal solutions; (C2) checking the existence of solutions is a decidable (ideally, tractable) problem; and (C3) for every source instance that has a solution, at least one universal solution can be computed (hopefully, in polynomial time).

The main algorithmic tool that the data exchange community has applied in order to check for the existence of solutions is the well-known *chase* procedure [37, 9], that was originally designed to reason about the implication problem for data dependencies. In data exchange, the chase is used as a tool for constructing a universal solution for a given source instance. The basic idea is the following. The chase starts with the source instance I, and then triggers every dependency in $\Sigma_{st} \cup \Sigma_t$ that is being violated, as long as this process is applicable. In doing so, the chase may fail (if firing an egd forces two constants to be equal) or it may never terminate (for instance, in some cases when the set of tgds is *cyclic*). It follows from [19], that if the chase fails then I has no solution; and that if the chase does not fail and terminates, then the resulting target instance is guaranteed to be a universal solution for I. Nothing can be said in the case when the chase does not terminate. The next example shows an application of the chase procedure.

Example 4.2 *Let $\mathcal{M}$ be the setting such that the source schema consists of the binary relation E, the target schema consists of the binary relations G and L, and Σ_{st} consists of the std $\varphi = E(x, y) \to G(x, y)$. Assume first that Σ_t consists of the tgd $\theta_1 = G(x, y) \to \exists z L(y, z)$, and let I be the source instance $E(a, b)$. The chase starts by firing φ and, thus, by populating the target with the fact $G(a, b)$. In a second stage, the chase realizes that θ_1 is being violated, and thus, θ_1 is triggered. The target is then extended with a fact $L(b, \perp)$, where $\perp$ is a fresh null value. At this stage, no dependency is being violated, and thus, the chase stops with result $J = \{G(a, a), L(b, \perp)\}$. It is not hard to see that J is a universal solution for I.*

Assume now that Σ_t is extended with the tgd $\theta_2 = L(x, y) \to \exists z G(y, z)$. Clearly, J does not satisfy θ_2 and the chase triggers this tgd. This means that a fact $G(\perp, \perp_1)$ is added to the target, where $\perp_1$ is a fresh null value. But θ_1 is now being violated again, and a new fact $L(\perp_1, \perp_2)$, where $\perp_2$ is a fresh null value, will have to be added. It is clear that this process will continue indefinitely, and thus, that the chase does not terminate.

Assume finally that Σ_t consists of the egd $\alpha = G(x, y) \to x = y$. Then the chase for I fails, since after populating the target instance with the fact $G(a, b)$, the egd α forces to equate the constants a and b. Notice that, in this case, I has no solution. □

As we have seen, the main problem with the application of the chase is non-termination. Fortunately, there is a meaningful class of data exchange settings, that is introduced next, for which the chase is guaranteed to terminate; further, it does so in at most polynomially many steps. It will follow that this class of settings satisfies our desiderata expressed above as conditions C1, C2 and C3.

Assume that Σ is a set of tgds over $\mathbf{T}$. We construct the *dependency* graph of Σ as follows. The nodes (positions) of the graph are all pairs (T, A), for $T \in \mathbf{T}$ and A an attribute of T. We add edges as follows. For every tgd $\forall \bar{x}(\varphi(\bar{x}) \to \exists \bar{y} \psi(\bar{x}, \bar{y}))$ in Σ, and for every $x \in \bar{x}$ that occurs in φ in position (T, A) and that also occurs in ψ, do the following:

- for every occurrence of x in ψ in position (S, B), add an edge from (T, A) to (S, B) (if the edge does not already exist); and
- for every existentially quantified variable $y \in \bar{y}$ and for every occurrence of y in ψ in position (R, C), add an edge labeled $\star$ from (T, A) to (R, C) (if the edge does not already exist).

Finally, we say that Σ is *weakly acyclic* if the dependency graph of Σ does not have a cycle going through an edge labeled $\star$.

Example 4.3 *Let θ_1 and θ_2 be as in Example 4.2. It is not hard to see that the set $\{\theta_1\}$ is weakly acyclic. On the other hand, the set $\{\theta_1, \theta_2\}$ is not.* □

Interesting classes of weakly acyclic sets of tgds include the following (see [30]): (1) Sets of tgds without existential quantifiers, and (2) *acyclic* sets of inclusion dependencies as defined in [13]. The notion of weak acyclicity was first formulated by Deutsch and Popa, and later independently used in [19] and [15]. The intuition behind this notion is as follows. Edges labeled $\star$ keep track of positions (R, C) for which the chase will have to create a fresh null value every time the left hand side of the corresponding tgd is triggered. Thus, a cycle through an edge labeled $\star$ implies that a fresh null value created in a position at a certain stage of the chase may determine the creation of another fresh null value, in the same position, at a later stage. Therefore, sets of tgds that are not weakly acyclic may yield non-terminating chase sequences (e.g. the set $\{\theta_1, \theta_2\}$). On the other hand, it has been proved in [19] that the chase always terminates for data exchange settings with a weakly acyclic sets of tgds. Further, in this case the chase for I terminates in at most polynomially many stages. This good behavior also implies the good behavior of this class of settings with respect to data exchange, as summarized below:

Theorem 4.4 *[19] Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st} \cup \Sigma_t)$ be a fixed data exchange setting, such that Σ_t is the union of a set of egds and a weakly acyclic set of tgds. Then there is a polynomial time algorithm such that for every source instance I, it first decides whether a solution for I exists, and if that is the case, it computes a universal solution for I in polynomial time.*

Thus, the class of settings with a weakly acyclic set of tgds satisfies conditions C1, C2, and C3, as defined above, and therefore, it constitutes a good class for data exchange according to our definition. In this case, the universal solution that can be constructed in polynomial time (if a solution exists at all) is precisely the result of the chase. This is usually called the *canonical* universal solution [19].

Let us mention that there are interesting classes of dependencies for which the problem of checking for the existence of solutions is trivial. For instance, by inspecting the proof of Theorem 4.4, one notices that for settings with a weakly acyclic set of tgds but without egds, source instances always have (universal) solutions. In particular, for settings without target dependencies it is the case that every source instance has at least one universal solution.

Other extensions Due to the advent of data exchange, the last years have seen a renewed interest on finding restrictions that guarantee termination of the chase. As we have just mentioned, one such restriction is the class of settings with a weakly acyclic set of tgds. However, both [16] and [38] showed that there are even broader classes of settings that preserve the good properties for data exchange. For some of these classes, the gain in expressive power comes at the cost of complexity: checking whether a setting belongs to some of these classes is in coNP, while it is clearly polynomial to verify whether a set of tgds is weakly acyclic.

In this section, the complexity analysis of the problem of checking the existence of solutions has been carried out assuming settings to be fixed. In other terms, we have studied the *data* complexity of the problem. While data complexity makes sense in a lot of data exchange scenarios, a more refined complexity analysis should consider both source instances and settings to be the input. This corresponds to the *combined* complexity of the problem. Kolaitis et al. have shown in [31] that the combined complexity of the problem of checking for the existence of solutions for settings with a weakly acyclic set of tgds is in EXPTIME, and can be EXPTIME-hard even if restricted to settings without tgds.

4.1 The core

Let us recall Examples 2.2 and 3.2. We mentioned that the instance $J = \{P(a, b, \perp_1), P(a, b, \perp_2), Q(\perp_3, \perp_1)\}$ is a solution for the source instance $\{M(a, b), N(a, b)\}$. Indeed, it is not hard to see that J is the canonical universal solution for I. Now, consider the instance $J^* = \{P(a, b, \perp_1), Q(\perp_3, \perp_1)\}$ that is contained in J. Then J^* is also a solution for I and, moreover, there is a homomorphism $h : J \to J^*$. Thus, J^* is also a universal solution.

We can draw an interesting conclusion from this example: among all possible universal solutions, the canonical universal solution is not necessarily the smallest (as J^* is strictly contained in J). Moreover, in the example, J^* is actually the smallest universal solution (up to isomorphism).

The first natural question is whether there is always a unique smallest universal solution. This was answered positively by Fagin et al. in [20]. The authors of [20] also argued that this smallest universal solution is the "best" universal solution, since it is the most economical one in terms of size, and that this solution should be the preferred one at the moment of materializing a solution. The whole issue is then how to characterize this smallest universal solution.

One of the main contributions in [20] is showing that the smallest universal solution always coincides with the *core* of the universal solutions. The *core* is a concept that originated in graph theory [28]; here we present it for arbitrary instances. Let K be an instance with values in $\texttt{Const} \cup \texttt{Var}$, and let K' be a subinstance of K. We say that K' is a *core* of K if there is a homomorphism from K to K' (recall that homomorphisms have to be the identity on constants), but there is no homomorphism from K' to a proper subinstance of itself. It is known that every instance has a core, and all cores of an instance are isomorphic [28, 20]. Thus, we can talk about *the* core of an instance K.

An instance J of $\mathbf{R}$ is a *subinstance* of I if the domain of J is contained in the domain of I and $J(R) \subseteq I(R)$, for every R in $\mathbf{R}$. If one of the inclusions is proper, we refer to J as a *proper subinstance* of I.

Example 4.5 *(Example 2.2 continued) The solution* $J^* = \{P(a, b, \perp_1), Q(\perp_3, \perp_1)\}$ *is the core of the universal solutions for* I*, since there is a homomorphism from* J *to* J^* *but there is no homomorphism from* J^* *to a proper subinstance of itself.* □

The next result summarizes some of the good properties of cores in data exchange.

Proposition 4.6 *[20] Let* $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st} \cup \Sigma_t)$ *be a setting. If* I *is a source instance and* J *is a universal solution for* I*, then the core of* J *is also a universal solution for* I *that is precisely the smallest universal solution.*

Thus, the core of the universal solutions has good properties for data exchange. This naturally raises the question about the computability of this core. As we have mentioned, the chase yields a universal solution that is not necessarily the core of the universal solutions, so different techniques have to be applied in order to compute this solution.

It is well-known that computing the core of an arbitrary graph is a computationally intractable problem. However, in data exchange we are interested in computing the core of a universal solution and not of an arbitrary instance, and the intractability of the former problem does not follow from the intractability of the latter. Indeed, it has been shown in [26] that computing the core of the universal solutions under the class of settings with a weakly acyclic set of tgds is a tractable problem.

Theorem 4.7 *[26] Let* $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st} \cup \Sigma_t)$ *be a fixed data exchange setting, such that* Σ_t *consists of a set of egds and a weakly acyclic set of tgds. There is a polynomial-time algorithm that for every source instance* I*, checks whether a solution for* I *exists, and if that is the case, computes the core of the universal solutions for* I*.*

A simple greedy algorithm that computes the core in polynomial time can be defined in the case when Σ_t consists of a set of egds [20, 5]. Unfortunately, this algorithm cannot be easily adapted to more complex settings, and much more sophisticated techniques have to be developed if one wants to prove that computation of cores of universal solutions continues to be a tractable problem in the presence of tgds. These techniques are based on the *blocks* method, that was also introduced in [20]. Gottlob in [25] developed a refined version of the blocks method, and proved with it that computing cores of universal solutions for settings whose set of target dependencies consist of egds and a set of tgds without existential quantifiers can be done in polynomial time. Later, in [26], Gottlob and Nash developed an even more sophisticated version of this method, and proved Theorem 4.7 with it.

5 Query Answering

Another big issue in data exchange is the semantics of query answering. Assume that a user poses a query Q over the target schema $\mathbf{T}$, and I is a given source instance. Then, what does it mean to answer Q with respect to I? Clearly, there is an ambiguity here, since there may be many solutions for I, and the evaluation of Q over different solutions may give different answers. The fact that one materializes only a single solution does not mean that others should not be taken into account when defining the semantics of the query.

There is a general agreement in the database community that in the context of data exchange and other related scenarios, like databases with incomplete information and data integration [32, 33], the right semantics is that of *certain* answers. Intuitively, an answer is *certain* if it occurs in the result of evaluation of query Q over every possible solution J. Notice that the semantics of a query is independent of a particular solution that is materialized.

Formally, let $\mathcal{M}$ be a data exchange setting, let Q be a query in some query language (typically FO), and let I be a source instance. We define $\texttt{certain}_{\mathcal{M}}(Q, I)$, the set of *certain answers of* Q *with respect to* I *under* $\mathcal{M}$, as $\bigcap\{Q(J) \mid J \text{ is a solution for } I\}$. We omit $\mathcal{M}$ if it is clear from the context. If Q is a query of arity 0 (a *Boolean* query), then $\texttt{certain}_{\mathcal{M}}(Q, I) = \texttt{true}$ iff Q is true in every solution J for I; otherwise, $\texttt{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$.

Example 5.1 *(Example 2.2 continued) The certain answers of the query* $Q = P(x, y, z)$ *with respect to* I *is the empty set. On the other hand, it is not hard to see that* $\texttt{certain}_{\mathcal{M}}(Q', I) = \{(a, b)\}$*, for* $Q' = \exists z P(x, y, z)$*.* □

Given a setting $\mathcal{M}$ and a query Q, the problem of computing certain answers for Q under $\mathcal{M}$ is, given a source instance I and a tuple $\bar{t}$, determine whether $\bar{t} \in \texttt{certain}_{\mathcal{M}}(Q, I)$. We deal with data complexity, i.e., assume that both $\mathcal{M}$ and Q are fixed. Finding certain answers involves computing the intersection of a (potentially) infinite number of sets. This strongly suggests that computing certain answers for arbitrary FO queries is an undecidable problem. Indeed, this is a rather straightforward consequence of Theorem 4.1. This does not preclude, however, the existence of interesting classes of queries for which the problem of computing certain answers is decidable, and even tractable. Indeed, next theorem shows that this is the case for the class of unions of conjunctive queries. Recall that a *conjunctive* query is an FO formula of the form $\exists \bar{x} \varphi(\bar{x}, \bar{y})$, where $\varphi(\bar{x}, \bar{y})$ is a conjunction of atoms.

Theorem 5.2 *[19] Let* $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st} \cup \Sigma_t)$ *be a data exchange setting, such that* Σ_t *consists of a set of egds and a weakly acyclic set of tgds, and let* Q *be a union of conjunctive queries. Then the problem of computing certain answers for* Q *under* $\mathcal{M}$ *can be solved in polynomial time.*

This is a very positive result, since unions of conjunctive queries are very common database queries: they correspond to the *select-project-join-union* fragment of relational algebra, and to the core of the standard query language for database systems, SQL. Theorem 5.2 can be easily proved when we put together the following three facts:

(1) Unions of conjunctive queries are preserved under homomorphisms, (2) every universal solution can be homomorphically mapped into any other solution, and (3) FO queries, and in particular, unions of conjunctive queries, have polynomial time data complexity.

Thus, in order to compute the certain answers to a union of conjunctive queries Q with respect to a source instance I it is sufficient to do the following. First, check whether a solution for I exists. If there is no solution, simply declare the setting to be inconsistent with respect to the given source instance. Otherwise, compute an arbitrary universal solution J for I. Finally, compute the set $\texttt{ground}(Q(J))$ of all those tuples in $Q(J)$ that do not contains nulls. From all the previous observations, and from the fact that tuples in the certain answers can only consist of constants, we derive that $\texttt{ground}(Q(J)) = \texttt{certain}_{\mathcal{M}}(Q, I)$. Notice that for the class of settings with a weakly acyclic set of tgds this algorithm runs in polynomial time. Finally, the algorithm also shows another desirable property of unions of conjunctive queries for data exchange; namely, their certain answers can be computed using a materialized target instance alone (e.g. the canonical universal solution or the core).

A mild, but interesting extension of the class of conjunctive queries is the class of conjunctive queries with inequalities. Unfortunately, this extension not only destroys preservation under homomorphisms but also leads to intractability of the problem of computing certain answers (first shown in [1] and then sharpened in [36]):

Theorem 5.3 *[36] There is a single Boolean conjunctive query Q with two inequalities and a setting $\mathcal{M}$ without target dependencies, such that the problem of computing certain answers for Q under $\mathcal{M}$ is* coNP*-complete.*

In particular, under widely believed complexity-theoretic assumptions, there is no polynomial-time algorithm that computes certain answers to conjunctive queries with inequalities using the canonical universal solution or the core, even in settings without target dependencies. On the other hand, by using techniques based on the chase procedure, Fagin et al. proved in [19] that the problem of computing certain answers to unions of conjunctive queries, with at most one inequality per disjunct, can be solved in polynomial time using an arbitrary universal solution.

Query rewriting In [4], Arenas et al. studied the following problem. Let Q be a FO query. Is it possible to find an FO query Q' such that for every source instance I, the certain answers of Q with respect to I correspond exactly to the evaluation of Q' over either the canonical universal solution or the core of the universal solutions for I? If that is the case, the query Q' is called a *rewriting* of Q over the corresponding universal solution. In particular, in [4] the authors developed techniques that helps to determine when a query admits a rewriting. This work also compared the canonical universal solution and its core in terms of the expressive power for allowing rewritings. It was shown that every query that admits a rewriting over the core of the universal solutions also admits a rewriting over the canonical universal solution, but not vice versa.

Extensions More expressive query languages, that allow for the presence of recursion and a restricted form of negation, and that preserve the good properties of unions of conjunctive queries for data exchange, were recently introduced by Arenas et al. [8]. Translations into this query language provide an alternative proof that computing certain answers for unions of conjunctive queries, with at most one inequality per disjunct, can be done in polynomial time. That paper also studied the combined complexity of computing certain answers in data exchange.

6 Alternative Semantics

The certain answers semantics is by no means unique, although it had been predominant in the early stages of data exchange research. In fact, the certain answers semantics has its problems, and a question "What is so sacred about the certain answers semantics?" was posed in [10]. Several attempts to address it were made; we survey them below.

6.1 Universal solutions semantics

Since [19], the seminal paper in data exchange, made a compelling case that the preferred solutions in data exchange should be the universal solutions, it seems natural to think of an alternative semantics that is completely based on those solutions. Formally, let $\mathcal{M}$ be a data exchange setting, let Q be a query, and let I be a source instance. We define the set of *universal certain answers to Q with respect to I under $\mathcal{M}$* denoted by $\texttt{certain}^u_{\mathcal{M}}(Q, I)$, as the set $\bigcap\{Q(J) \mid J \text{ is a universal solution for } I\}$. The problem of computing the universal certain answers to Q under $\mathcal{M}$ is defined analogously to the case of the standard semantics.

Since computing certain answers for a union of conjunctive queries can be done by posing the query over an arbitrary universal solution, it easily follows that for each query Q of that form and every source instance I, $\texttt{certain}_{\mathcal{M}}(Q, I) = \texttt{certain}^u_{\mathcal{M}}(Q, I)$. But this is no longer the case if we allow inequalities.

Example 6.1 *(Example 2.2 continued) The universal certain answers to $Q = \exists x \exists y \exists z (P(x, y, z) \wedge x \neq z)$ with respect to I is* `true`*, since every universal solution J for I must contain a fact of the form $P(a, b, \perp)$, for $\perp$ a value in* `Var`*. On the other hand,* $\texttt{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$ *since $J = \{P(a, b, a), Q(b, a)\}$ is a solution for I.*

We show next that the universal certain answers semantics presents some computational advantages over the usual semantics. An *existential* FO formula is a formula of the form $\exists \bar{x} \varphi(\bar{x}, \bar{y})$, where $\varphi(\bar{x}, \bar{y})$ is a quantifier-free formula

in disjunctive normal form. Each existential FO formula θ is known to have the following property: If K_1 is an induced subinstance of K_2 and $\bar{t}$ belongs to the evaluation of θ over K_1, then $\bar{t}$ belongs to the evaluation of θ over K_2. Since every universal solution contains an isomorphic copy of the core as an induced subinstance, it easily follows that the problem of computing the universal certain answers for an existential FO query θ boils down to evaluating θ over the core of the universal solutions, discarding all tuples that do not consist only of constants. Putting this together with Theorem 4.7 it is possible to obtain the following:

Theorem 6.2 *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st} \cup \Sigma_t)$ be a setting, such that Σ_t consists of a set egds and a weakly acyclic set of tgds, and let Q be a (domain-independent) existential* FO *formula. Then the problem of computing universal certain answers of Q under $\mathcal{M}$ can be solved in polynomial time.*

Notice that this is in deep contrast with Theorem 5.3 that shows that for very simple existential FO formulas (namely, conjunctive queries with two inequalities) the problem of computing certain answers becomes intractable.

6.2 Semantics based on the closed-world assumption and incomplete information

Both the certain answers and the universal certain answers semantics are based on an *open-world assumption* (OWA), i.e. solutions included in those semantics are open to adding new facts. This assumption, however, leads to some anomalies with respect to query answering, as described next. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a data exchange setting without target dependencies. We say that $\mathcal{M}$ is *copying* if $\mathbf{S} = \langle R_1, \ldots, R_m \rangle$, $\mathbf{T} = \langle R'_1, \ldots, R'_m \rangle$, and Σ_{st} consists of the stds $\forall \bar{x}(R_i(\bar{x}) \rightarrow R'_i(\bar{x}))$ (that is, R_i and R'_i have the same arity). It is pointed out in [4] that there is a copying data exchange setting $\mathcal{M}$ and an FO query Q over the target schema, such that Q is not FO rewritable over the canonical universal solution under $\mathcal{M}$. That is, there is no FO query Q' such that for every source instance I, the evaluation of Q' over the canonical universal solution J for I coincides with the certain answers of Q with respect to I. This behavior is strange, as intuitively copying settings only change relation names and canonical universal solutions are just copies of source instances. As such, one would expect every FO query to be rewritable by itself over the canonical universal solution.

The previous anomaly arises because OWA assumes the canonical universal solution to be open to adding new facts – both under the certain answers and the universal certain answers semantics – while intuitively we would expect the canonical universal solution to be the only solution under copying settings. Because of that, Libkin proposed in [34] that the right assumption in data exchange should not be the OWA but its usual competitor, the *closed-world assumption* (CWA), in which solutions are closed to adding new facts. The reason behind this is that if data exchange is about transferring data from source to target, then the semantics for the data exchange problem should be based on the exchanged data only, and not on data that can be later added to the instances. Or in other terms, the semantics should be based on those solutions that contain no more than what is needed to satisfy the specification.

Unfortunately, defining CWA-solutions in data exchange is not so easy as defining the OWA solutions. Here we avoid the rather elaborate operational definition of CWA-solutions given in [34], and provide a more elegant semantic description of this class (which appears as a characterization of such solutions in [34]). As usual, K' is a *homomorphic image* of K if there is a homomorphism $h : K \rightarrow K'$ such that $h(K)$, the instance obtained from K by replacing each null $\perp$ by $h(\perp)$, is exactly K'.

Definition 6.3 *Let $\mathcal{M}$ be a setting without target dependencies, I a source instance and J a solution for I. Then J is a* CWA-solution *for I if (1) J is a universal solution for I, and (2) J is a homomorphic image of the canonical universal solution for I.*

Thus, if $\mathcal{M}$ is a copying setting and I is a source instance, then the unique CWA-solution for I under $\mathcal{M}$ is its canonical universal solution J (since any other universal solution is not a homomorphic image of J). Next we show a slightly more interesting example.

Example 6.4 *(Example 2.2 continued) Recall that $I = \{M(a,b), N(a,b)\}$. Then, with respect to I, both its canonical universal solution $J = \{P(a,b,\perp_1), P(a,b,\perp_2), Q(\perp_3,\perp_1)\}$ and the core of its universal solutions $J^* = \{P(a,b,\perp_1), Q(\perp_3,\perp_1)\}$ are* CWA-*solutions. On the other hand, the solution $J_1 = \{P(a,b,\perp_1), Q(\perp_1,\perp_1)\}$ is not a* CWA-*solution, simply because it is not a universal solution for I. The solution $J_2 = \{P(a,b,\perp_1), P(a,b,\perp_2), P(a,b,\perp_4), Q(\perp_3,\perp_1)\}$ is a universal solution for I that it is not a homomorphic image of J. Therefore, J_2 is not a* CWA-*solution.* □

Notice the following properties of the class of CWA-solutions. The core of the universal solutions is a "minimal" element of the class, in the sense that every other CWA-solution contains an isomorphic copy of the core as a subinstance. Also, the canonical universal solution is in a way the "maximal" element of the class, as any other CWA-solution is a homomorphic image of it. This observation will be relevant later once we study the properties of query answering.

In addition to defining the class of CWA-solutions, [34] also points out that while target instances in data exchange are tables with nulls, techniques for handling incomplete information over target instances had been completely ignored in previous data exchange literature. Indeed, the usual semantics of data exchange (e.g. certain answers or universal certain answers) are defined over sets of solutions as if each one were a table without nulls.

But already 25 years ago, Imielinski and Lipski [32] showed that answering queries over databases with nulls must be done with care, and that treating nulls in the same way as constants yields semantically incorrect answers. The basic idea in [32] is that a database T with nulls represents a set $\mathbf{Rep}(T)$ of "complete" databases. i.e., a set of databases without nulls. We formally define this as follows. A *valuation* is a mapping $\nu : \mathtt{Var} \to \mathtt{Const}$. If T is an instance with elements in $\mathtt{Const} \cup \mathtt{Var}$, then $\nu(T)$ represents the instance obtained from T by replacing each null $\perp$ with $\nu(\perp)$. Notice that all the elements in $\nu(T)$ are constants. Then we define $\mathbf{Rep}(T) = \{\nu(T) \mid \nu \text{ is a valuation}\}$.

In order to evaluate a query Q over an incomplete database T, the standard approach is to compute the set $\Box Q(T) = \bigcap\{Q(D) \mid D \in \mathbf{Rep}(T)\}$. This set is usually called the certain answers of Q with respect to T in the incomplete information literature, but we prefer to represent it by $\Box Q(T)$ here, in order to avoid confusion with the certain answers as defined for data exchange.

Although [34] proposes four different semantics for data exchange, here we concentrate on one that seems to be the most relevant for data exchange and that is, at the same time, the closest to the semantics we have seen so far. Let $\mathcal{M}$ be a setting without target dependencies, Q an FO query, and I a source instance. Then we define the *certain answers under* CWA *and incomplete information* of Q with respect to I (under $\mathcal{M}$), denoted by $\mathtt{certain}^{\mathrm{CWA}}_{\mathcal{M}}(Q, I)$, as the set of tuples that would be in $Q(D)$ for every CWA-solution J and every $D \in \mathbf{Rep}(J)$, that is, $\bigcap\{\Box Q(J) \mid J \text{ is a CWA-solution for } I \text{ under } \mathcal{M}\}$.

Notice that for each source instance I with canonical universal solution J, and for every CWA-solution J' for I, it must be the case that $\mathbf{Rep}(J') \subseteq \mathbf{Rep}(J)$. Thus, we have that $\Box Q(J) \subseteq \Box Q(J')$, for every FO query Q. Hence:

Theorem 6.5 *[34] Let $\mathcal{M}$ be a setting without target dependencies and Q an arbitrary query. Then $\mathtt{certain}^{\mathrm{CWA}}_{\mathcal{M}}(Q, I) = \Box Q(J)$, for every source instance I with canonical universal solution J.*

That is, in this case the problem of evaluating certain answers under CWA and incomplete information boils down to the problem of evaluating queries over incomplete databases (canonical universal solutions).

Example 6.6 *(Example 2.2 continued) Consider Q_1 be a query asking whether the interpretation of relation Q has exactly one tuple. It can be shown that $\mathtt{certain}^{\mathrm{CWA}}_{\mathcal{M}}(Q_1, I) = \mathtt{true}$. Indeed, every* CWA*-solution must contain at least one tuple in the interpretation of Q, but it cannot contain two tuples: this is because every* CWA*-solution is a homomorphic image of the canonical universal solution $J = \{P(a, b, \perp_1), P(a, b, \perp_2), Q(\perp_3, \perp_1)\}$ for I. On the other hand, $\mathtt{certain}_{\mathcal{M}}(Q, I) = \mathtt{certain}^{u}_{\mathcal{M}}(Q, I) = \mathtt{false}$.* □

The previous example shows that there is a setting $\mathcal{M}$ and a FO query Q such that $\mathtt{certain}^{\mathrm{CWA}}_{\mathcal{M}}(Q, I) \neq \mathtt{certain}_{\mathcal{M}}(Q, I)$ for some source instance I. On the other hand, there is an interesting class of queries for which the semantics based on CWAand incomplete information coincides with the usual semantics. This is the class of *monotone* FO queries. It follows that the problem of computing certain answers under CWAand incomplete information for conjunctive queries with inequalities is coNP-complete, and that for unions of conjunctive queries it becomes tractable. For arbitrary FO queries the problem is always in coNP, which contrasts sharply with the case of the usual semantics where the problem of computing certain answers may be undecidable.

Extensions Recently, Hernich and Schweikardt extended the notion of CWA-solutions and semantics based on incomplete information to data exchange settings with target dependencies [29]. Further, Libkin and Sirangelo [35] have recently proposed a mixed approach to data exchange that combines the closed- and the open-world assumption.

7 Related Work and Extensions

Data exchange is closely related to the problem of data integration [33]. A data integration system consists of a *local* schema, a *global* schema, and a specification of the relationship between the local and the global schema. The data resides at the local level, but the user can only query the data at the global level. Thus, a data integration system can be seen as a data exchange setting where the source schema corresponds to the local schema, and the target schema corresponds to the global schema. Furthermore, the goal in both data integration and data exchange is to compute the certain answers to a query that is posed over the target (global) schema.

The main difference between data exchange and *virtual* data integration is that in the latter, the data is never actually exchanged, as the global database corresponds only to a virtual representation of the local database. As such, the goal in data integration is to compute the certain answers to a query based on the local data, as opposed to data exchange where the goal is to do the same but using a materialized target instance. But there are many commonalities, and the formal study of the relationship between data exchange and data integration deserves further study (see, e.g., [14]).

An area of study that has evolved from data exchange, but that by now has become a prominent topic on its own, is schema management. The fundamental idea is that schema mappings correspond to metadata, and that it is important to have a conceptual framework in which this metadata is combined by applying some predefined operators [11]. The formal study of two of these operators, the *composition* and the *inverse*, has recently been started [18, 21, 22, 7]. The interesting part of this story for data exchange is that, in many cases, in order to completely understand the schema management operator under scope, it is necessary to use more expressive source-to-target dependencies than the ones used

in this article. This defines a completely new data exchange problem.

Several extensions of the data exchange problem, as studied here, have been proposed in the recent years. For instance, Fuxman et al. proposed in [24] an extension of the class of data exchange settings in which dependencies from target-to-source are also allowed. The motivation for studying this class of settings comes from the area of peer data management systems [27], which model the case when different databases (peers) interact with each other, sharing and exchanging data. The source peer is the "authoritative" peer that contributes with data. On the other hand, the target peer restricts the data it is willing to accept by means of target-to-source dependencies, but has no right to modify the source data.

Other extensions include the class of queries that data exchange systems can support and the class of values allowed in source instances. With respect to the first one, Afrati and Kolaitis recently studied the topic of answering aggregate queries in data exchange [2]. In order to do so, they had to define a new semantics for query answering, based on the class of endomorphic images of the canonical universal solutions. This is because all the semantics we have seen so far yield rather trivial semantics for aggregate queries in data exchange. With respect to the latter, Fagin et al. [23] proposed an extension of data exchange that allows for null values also to appear in source instances. This represents the fact that incomplete information can also arise in the source; e.g. when the materialized target instance of one data exchange setting is used as the source instance of another setting.

Finally, in this article we studied the logical foundations of relation data exchange. However, semistructured models, and in particular, XML, have been specially designed for exchanging data on the web. Thus, it seems natural to develop a notion of XML data exchange. The foundational study of this problem was initiated by Arenas and Libkin [6], which unveiled many of the fundamental problems that appear in this new scenario; schema mappings for XML were studied in Amano et al. [3].

8 Concluding Remarks

Data exchange is nowadays one of the most active areas of database theory. In this paper we tried to give a brief introduction to the main lines of research involved in this problem, with a special emphasis on the topics of materializing solutions, query answering, and the notions of the semantics for data exchange.

Some of these topics are already quite mature, while others need further exploration. For instance, we have mentioned that doing proper query answering is one of the main goals of data exchange systems, but nevertheless, the knowledge we have about this topic is rather limited and several interesting questions remain open. One of the most challenging is finding bigger classes of database queries for which the problem of computing certain answers is decidable. Also, we believe that the time is ripe for having a fruitful discussion that compares the different semantics proposed for query answering in data exchange, and developing criteria that help determine when a semantics is appropriate. A natural question, for instance, is whether we should have an overall data exchange semantics that works for every query, or whether it would be better to take a more pragmatic view that takes into account the class of queries at hand, and permits a lower quality of the answer in favor of the performance of the system.

As we have mentioned, developing a systematic study of XML data exchange is crucial for understanding the transfer of data on the web. However, there are very few papers that have dealt with this issue up till now. Relevant, but unexplored topics related to XML data exchange include, for example, the problems of existence of solutions, materializing solutions, query answering beyond conjunctive queries, and notions of CWA and incomplete information.

Acknowledgment The author gratefully acknowledges support by FONDECYT grant 11080011. Also, comments by M. Arenas and L. Libkin on an early draft have been of great help.

References

[1] S. Abiteboul, and O. Duschka. Answering queries using materialized views. Gemo report 383.

[2] F. N. Afrati, and P. G. Kolaitis. Answering aggregate queries in data exchange. In *PODS*, pages 129–138, 2008.

[3] S. Amano, L. Libkin, and F. Murlak. XML schema mappings. In *PODS*, 2009.

[4] M. Arenas, P. Barceló, R. Fagin, and L. Libkin. Locally consistent transformations and query answering in data exchange. In *PODS*, pages 229–240, 2004.

[5] M. Arenas. Personal communication, 2004.

[6] M. Arenas, L. Libkin. XML data exchange: Consistency and query answering. *Journal of the ACM* 55(2), 2008.

[7] M. Arenas, J. Pérez, C. Riveros. The recovery of a schema mapping: bringing exchanged data back. In *PODS*, pages 13-22, 2008.

[8] M. Arenas, P. Barceló, J. Reutter. Query languages for data exchange: Beyond unions of conjunctive queries. In *ICDT*, 2009.

[9] C. Beeri, and M. Y. Vardi. A proof procedure for data dependencies. *Journal of the ACM*, 31(4):718–741, 1984.

[10] L. Bertossi, J. Chomicki, P. Godfrey, P. Kolaitis, A. Thomo, C. Zuzarte: Exchange, integration, and consistency of data: report on the ARISE/NISR workshop. SIGMOD Record 34(3): 87-90, 2005.

[11] P. Bernstein. Applying model management to classical meta-data problems. In *CIDR*, 209-220, 2003.

[12] Clio Project. www.almaden.ibm.com/software/projects/criollo

[13] S. Cosmadakis, P. Kanellakis. Functional and inclusion dependencies; A graph theoretica approach. In *Advances in Computing Research*, vol. 3, 163.184, 1986.

[14] G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati. On reconciling data exchange, data integration, and peer data management. In *PODS*, pages 133–142, 2007.

[15] A. Deutsch, V. Tannen. Reformulation of XML queries and constraints. In *ICDT*, 225-241, 2003.

[16] A. Deutsch, A. Nash, and J. B. Remmel. The chase revisited. In *PODS*, pages 149-158, 2008.

[17] R. Fagin. Horn clauses and database dependencies. *Journal of the ACM*, 29(4): 952-985, 1982.

[18] R. Fagin, P. Kolaitis, L. Popa, W. C. Tan. Composing schema mappings: Second-order dependencies to the rescue. In *PODS*, pages 83–94, 2004.

[19] R. Fagin, P. G. Kolaitis, R. J. Miller, L. Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005. Preliminary version in *ICDT*, 2003.

[20] R. Fagin, P. G. Kolaitis, and L. Popa. Data exchange: getting to the core. *ACM Transactions on Database Systems*, 30(1):174–210, 2005.

[21] R. Fagin. Inverting schema mappings. In *PODS*, pages 50-59, 2006.

[22] R. Fagin, P. Kolaitis, L. Popa, W.-C. Tan. Quasi-inverses of schema mappings. *ACM Transactions on Database Systems* 33(2), (2008).

[23] R. Fagin, P. Kolaitis, L. Popa, W.-C. Tan. Reverse data exchange: Coping with nulls. In *PODS*, 2009.

[24] A. Fuxman, P. Kolaitis, R. Miller, W.-C. Tan. Peer data exchange. In *PODS*, pages 160-171, 2005.

[25] G. Gottlob. Computing cores for data exchange: New algorithms and practical solutions. In *PODS*, pages 148–159, 2005.

[26] G. Gottlob, A. Nash. Data exchange: Computing cores in polynomial time. In *PODS*, pages 40–49, 2006.

[27] A. Halevy, Z. Ives, D. Suciu, I. Tatarinov. Schema mediation in peer data management systems. In *ICDE*, pages 505-518, 2003.

[28] P. Hell, J. Nešetřil. *Graphs and homomorphisms*. Oxford University Press, 2004.

[29] A. Hernich, N. Schweikardt. CWA-solutions for data exchange settings with target dependencies. In *PODS*, pages 113-122, 2007.

[30] P. Kolaitis. Schema mappings, data exchange, and metadata management. In *PODS*, pages 61–75, 2005.

[31] P. Kolaitis, J. Panttaja, and W.-C. Tan. The complexity of data exchange. In *PODS*, pages 30–39, 2006.

[32] T. Imielinski, W. Lipski. Incomplete information in relational databases. *Journal of the ACM* 31, 761–791, 1984.

[33] M. Lenzerini. Data integration: A theoretical perspective. In *PODS*, pages 233–246, 2002.

[34] L. Libkin. Data exchange and incomplete information. In *PODS*, pages 60–69, 2006.

[35] L. Libkin, C. Sirangelo. Data exchange and schema mappings in open and closed worlds. In *PODS*, pages 139–148, 2008.

[36] A. Mądry. Data exchange: On the complexity of answering queries with inequalities. *Information Processing Letters*, 94(6):253–257, 2005.

[37] D. Maier, A. Mendelzon, Y. Sagiv. Testing implications on data dependencies. *ACM Transactions on Database Systems*, 4(4), 455-469, 1979.

[38] M. Meier, M. Schmidt, G. Lausen. Stop the Chase. In *Alberto Mendelzon Workshop*, AMW, 2009.

[39] Second International Workshop on Exchange and Integration of Data, EIS06. http://www.scs.carleton.ca/ diis/EID06/

Data Integration in Mashups

Giusy Di Lorenzo
Dipartimento di Informatica e Sistemistica
University of Naples Federico II, Via Claudio, 21, 80125 Napoli, Italy
giusy.dilorenzo@unina.it

Hakim Hacid*
Alcatel-Lucent Bell Labs France
Centre de Villarceaux, 91620 Nozay
hakim.hacid@alcatel-lucent.com

Hye-young Paik, Boualem Benatallah
CSE, UNSW
Sydney, NSW 2052, Australia
hpaik@cse.unsw.edu.au
boualem@cse.unsw.edu.au

ABSTRACT

Mashup is a new application development approach that allows users to aggregate multiple services to create a service for a new purpose. Even if the Mashup approach opens new and broader opportunities for data/service consumers, the development process still requires the users to know not only how to write code using programming languages, but also how to use the different Web APIs from different services. In order to solve this problem, there is increasing effort put into developing tools which are designed to support users with little programming knowledge in Mashup applications development. The objective of this study is to analyze the richnesses and weaknesses of the Mashup tools with respect to the data integration aspect.

1. INTRODUCTION

One of the goals of Web 2.0 is to make it easy to create, use, describe, share, and reuse resources on the Web. To achieve that, technologies have flourished around this concept (e.g., blogs, social networks). The capabilities of Web 2.0 are further enhanced by many service providers who expose their applications in two ways: one is to expose application functionalities via Web APIs such as Google Map[1], Amazon.com, or Youtube[2], the other is to expose data feeds such as RSS and ATOM. This opened up new and exciting possibilities for service consumers and providers as it enabled the notion of using these *services*[3] as "ingredients" that can be mixed-and-matched to create new applications.

To achieve this goal, and maybe to anticipate future needs in Web 2.0, a new framework, called *Mashup*, is surfacing [12, 6, 21, 4]. *Mashup* is an application development approach that allows users to aggregate multiple services, each serving its own purpose, to create a service that serves a new purpose. Unlike Web services composition where the focus is on the composition of business (process) services only, the Mashup framework goes further in that it allows more functionalities and can compose heterogeneous resources such as data services, UI services, etc. Applications built using the Mashup technique are referred to as *Mashups* or *Mashup applications*. Their recent proliferation demonstrates that there is high level of interest in the Mashup framework [5, 14, 28, 18, 19]. It also shows that the needs for integrating these rich data and service sources are rapidly increasing. The Mashup approach opens new and broad opportunities for data/services consumers. However, the development investment is still considerable. In fact, a Mashup user needs to know, in addition to how to write code using programming languages (e.g., Java Script, XML/HTML), how to use the different Web APIs[4]. In order to solve this problem, there is an increasing effort put into developing tools which are designed to support users with little programming knowledge in Mashup applications development.

The objective of this study is to analyze the richnesses and weaknesses of the Mashup tools. Thus, we identify the behaviors and characteristics of general Mashup applications and analyze the tools with respect to the data integration aspect. We believe that this kind of study is important to drive future contributions in this emerging area where a lot of research and application fields, such as databases, user machine interaction, etc. can meet. Other studies, focusing on other aspects like process integration, interface integration [11] complement the work presented in this paper to understand different aspects of Mashups.

The remainder of this paper is organised as follows: Section 2 introduces the different levels of Mashups. Section 3 discusses the dimensions of analysis, and describes different Mashup tools according to the considered dimensions[5]. We finish by a discussion and a conclusion in Section 4, summarizing our study and highlighting some future directions.

*This work has been mainly done when the author was a Research Associate at UNSW.
[1]http://maps.google.com/
[2]http://youtube.com
[3]These services can be a data service, such as news, or a process/operation service such as placing an order to Amazon.com.

[4]A lot of APIs are available on the Web: http://www.programmableweb.com/apis/directory/
[5]We have selected these tools not because they are the best or the worst tools, but we have tried to consider as much representative tools as possible.

2. DESCRIPTION OF MASHUP LEVELS

One of the most important characteristics of the Web is certainly its heterogeneity. This heterogeneity can be seen on data, processes, and even user interfaces. Conceptually, a Mashup application is a Web application that combines information and services from multiple sources on the Web. Generally, web applications are developed using the *Model-View-Controller* pattern (MVC)[25] which allows the separation of core business model from the presentation and control logic which use the business model. In the MVC pattern, the *model* represents the data on which the application operates and the business rules used to manipulate the data. The model is independent of the view and the controller. It passively supplies its services and data to the other layers of the application. The *view* represents the output of the application. It specifies how the data, accessed through the model, is presented to the user. Also, it has to maintain its presentation when the model changes. Finally, the *controller* represents the interface between the model and the view. It translates interactions with the view into actions to be performed on the model.

A Mashup application includes all the three components of the MVC pattern. According to Maximilien et al. [23], the three major components of a Mashup application are (1) data level, (2) process level, and (3) presentation level. Moreover, each data source needs to be first analyzed and modeled in order to perform the required actions of retrieval and pre-processing. For the completeness of the study, we first describe those levels. We will then focus our analysis specifically on the data level.

1. *Data Level*: this level mainly concerns data mediation and integration. Challenges at this level involve accessing and integrating data residing in multiple and heterogeneous sources such as web data and enterprise data[17]. Generally, these resources are accessible through REST[15] or SOAP[6] web services, HTTP protocol and XML RPC. Regarding the data mediation, the basic problem comes from structural and semantics diversities of the schema to be merged[17][7]. Finally, data sources can be either structured for which a well defined data model is available (e.g., XML-based document, RSS/ATOM feed), or unstructured (e.g., audio, email text, office documents). In the latter case, the unstructured data needs to be pre-processed in order to extract their meaning and create structured data. So, this level consists of all possible data manipulations (conversion, filtering, format transformation, combination, etc.) needed to integrate different data sources. Each manipulation could be done by analyzing both syntax and semantics requirements.

2. *Process Level*: the integration at the process level has been studied specially in the workflow and service oriented composition areas [13][3]. At the *process level*, the choreography between the involved applications is defined. The integration is done at the application layer and the composed process is developed by combining activities, generally exposed through APIs. The composition is then described using either programming languages such as Java, or dedicated workflow languages such as WS-BPEL [3]. In the Mashups context, those languages are not enough for modeling applications since, for instance, they do not provide the connection to different remote resources, e.g., REST resources, and do not handle the interaction with the client browsers. These limitations make it difficult to directly use these technologies for Mashups. Thus, they need to be adapted in order to model and describe interactive and asynchronous processes. Currently, languages like *Bite*[10] or *Swashup*[23] have been proposed to describe the interaction and the composition model for Mashups.

3. *Presentation Level*: every application needs an interface to interact with the users, and a Mashup application is not an exception. Presentation Level (or User Interface) in Mashup applications is used to elicit user information as well as to display intermittent and final process information to the user. The technologies used to display the result to the user can be as simple as an HTML page, or a more complex web page developed with Ajax, Java Script, etc. The languages for integrating UI components and visualising the front-ends support server-side or client-side Mashups[1][2]. In a server-side Mashup, the integration of data and services is made on the server. The server acts as a proxy between the Mashup application and other services involved in the application. On the other hand, a client-side Mashup integrates data and services on the client. For example, in a client-side Mashup, an Ajax application will do the required composition and parse it into a client's web browser. Currently, the integration at the presentation level in Mashups is done manually. That is, a developer needs to combine the user interface of the wished components using either server-side or client-side technologies. This area is an emerging area and lot of efforts are made in this direction [11][29]. From the Mashup point of view, there is still a lot of work to be done.

In the next section, we introduce the dimensions used in our analysis. As mentioned before, we focus on the data level in Mashups. Other studies focusing on different aspects like presentation level (e.g.,[11]) are necessary and complement our study.

3. ANALYSIS DIMENSIONS

To understand why some automatic support is needed to create Mashups, we give the following example. Suppose that a user wants to implement a *News Mashup* that lets her select news on a news service like CNN International and display both the list of the news and a map that highlights the locations associated with the news; she typically needs to do a lot of programming which involves fetching and integrating heterogeneous data. In fact, the user needs to know not only how to write the code, but also (i) to understand the available API services in order to invoke them and fetch the data output; (ii) to implement screen scraping techniques for the services that do not provide APIs, and (iii) to know the data structure of the input and output of each service in order to implement a data mediation solution.

[6]Simple Object Access Protocol, http://www.w3.org/TR/soap/

The Mashup tools provide facilities to help the user solve some of the above mentioned problems. The analysis provided in this section aims at studying how the tools (see Section 3.1 for the list of concerned tools) address the data mediation problems discussed in the previous section. We will be asking questions such as: How the tools handle data? What kind of processing is performed on the data? What is the output of Mashups? Which operators are provided for data transformation and for creating the data flow? What are the types of data supported by the available operators? etc.

3.1 Analyzed Tools

Currently a number of Mashup tools exist. We have selected only the following tools for three main reasons: (i) these tools were the most popular ones when this analysis was performed (judging from discussions on forums, blogs, etc.), (ii) some other tools have not been reported because of their unavailability at certain stages of the study preventing us to experiment and report results according to our analysis. Finally, (iii) this limitation is motivated by the fact that our objective is not to analyze all the tools but to give a view on the current state of these tools and understand their general approach to data integration.

1. *Damia[5, 27]* is a Mashup tool provided by *IBM*. It allows the users to assemble data feeds from Internet and enterprise data sources. This tool focuses on data feed aggregation and transformation in enterprise environments. Additional tools or technologies like *QEDWiki*[7] and feed readers, that consume Atom and RSS, can be used at the presentation layer for the data feed provided by *Damia*.

2. *Yahoo pipes*[8] is a web-based tool provided by *Yahoo*. The users can build mashup applications by aggregating and manipulating data from web feeds, web pages, and other services. A pipe is composed of one or more modules, each one performing a single task like retrieving feeds from a web source, filter, sort or merge feeds. The output from pipes can be either accessed by a client via a unique URL as RSS or JSON, or visualised on the Yahoo Map.

3. *Popfly*[9] is a web-based Mashup application by Microsoft. It allows the users to create a Mashup combining data and media sources. The Mashup is built by connecting *blocks*. Each block is associated to a service like "Flicker"[10] and exposes one or more functionalities. *Popfly* is much more about data visualization than data manipulation as we will see later, so the "mashed" data can be only visualized using the provided visualization tool.

4. *Google Mashup Editor*(GME)[11] is a Mashup development, deployment and distribution environment by *Google*. The Mashup can be created using technologies like HTML, Java Script, CSS along with the GME XML tags and Java Script API that further allows a user to customize the presentation of the Mashup output.

5. *Exhibit*[18] is a framework for creating web pages containing dynamic and rich visualizations of structured data. Generally, it is used in combination with Babel[12]. It allows users to assemble data obtained in different format such as RDF/XML, N3, Bibtex. *Exhibit* visualises the mashup output on HTML pages, but it also lets the user access the data by explicitly exporting it to various formats including RDF/XML and Exhibit JSON.

6. *Apatar*[13] is a Mashup data integration tool that helps users integrate desktop data with the web. Users install a visual job designer application to create integration schemas called DataMaps. *Apatar*, like *DAMIA*, mainly aims to aggregate and manipulate data that can be reused from other applications, so additional tools that consume the *Apatar* output formats can be used as the presentation layer.

7. *MashMaker*[14] is a web-based tool by *Intel* for editing, querying and manipulating web data. *MashMaker* is different from the other tools in that it works directly on web pages. In fact, *MashMaker* allows users to create a mashup by browsing and combining different web pages. The final goal of this tool is to suggest to the user some enhancements (mashups or widgets), if available, for the visited web pages.

For the analysis, we chose eight dimensions covering various aspects of the data level concerns. Due to space limitations, we only discuss some of the above tools for each dimension for illustration purposes. A summary of the complete analysis is given in Table 1 and a detailed description is available in [22]. The eight dimensions are described in the following.

3.2 Data Formats and Access

In a Mashup application, a user can integrate data described in different formats. For example, web feed format is used to publish frequently updated content such as blog entries, news and so on; tabular format is suitable for describing table-based data models such as csv files or spreadsheets; markup-based format (e.g., HTML and XML) is, of course, commonly used to publish data; multimedia content such as video, audio and images are becoming increasingly prevalent. These types of data can be available to the user from different data sources. The most common data sources can be traditional database systems, local files that are available in the owner's file system, web pages, web services and web applications. To facilitate Web data retrieval, providers often expose their content through web APIs. APIs can be also seen as a useful means for data and application mediation. Here we consider the role of APIs from the data integration point of view in the sense that they offer specific types and formats of data. It should be noted that an API can offer several formats of data, e.g., csv, xml, etc.

[7] http://services.alphaworks.ibm.com/qedwiki/
[8] http://pipes.yahoo.com/pipes/
[9] *http : //www.popfly.net/*
[10] www.flickr.com/
[11] http://code.google.com/gme/index.html
[12] http://simile.mit.edu/babel/
[13] www.apatar.com/

Table 1: Summary of the considered dimensions. (+) means the dimension (i.e. functionality) is provided, (-) means the dimension is not provided. These marks do not imply any "positive" or "negative" information about the tools except the presence or the absence of the considered dimension.

		Damia	Yahoo Pipes	MS Popfly	GME	Exhibit	Apatar	MashMaker
Data Formats/Access	Protocol[a]	P_2	P_2	P_2 ,P_3	P_2	P_2	P_1, P_2	P_1
	Data Format[b]	D_1, D_2 D_7, D_8	D_1, D_2, D_3, D_4, D_5, D_6, D_8, D_9	D_1, D_2, D_9, D_{10}	D_2, D_3, D_4	D_1, D_4, D_6 D_7, D_8, D_9	D_1, D_2, D_6 D_7, D_9	D_5
	Database [c]	DB_1, DB_2[d]	-	-	-	-	DB_3, DB_4, DB_5	-
Internal Data Model	XML-based	+	+	-	+	-	-	+
	Object-based	-	-	+	-	+	+	-
Data Mapping	Manual	+	-	+	-	-	+	+
	Semi-Automatic	-	+	-	-	-	-	-
	Automatic[e]	-	-	+	+	+	-	-
Data Refresh	Pull strategy	+	+	+	+	+	+	+
	Push strategy	-	-	-	-	-	-	-
	Global pull interval	-	+	+	+	+	+	+
	Local pull interval	+	-	-	-	-	-	-
	Interval Setting	+	-	-	-	-	-	-
Mashup's Output	Machine oriented	+	+	-	-	+	+	-
	Human oriented	-	+	+	+	+	-	+
Extensibility	Components	+	+	+	+	+	+	+
	Data	-	-	-	-	-	-	+
Sharing	Total	+	+	+	+	-	+	+
	Partial	-	+	-	-	-	+	-
	No thing	+	+	+	+	-	+	+
	Read only	+	+	+	+	-	+	+
	Read/Write	-	-	-	+	-	-	-
	All users	+	+	+	+	-	+	+
	Groups	-	-	-	+	-	-	-
	Particular user	+	+	+	+	-	+	+

[a] P_1 = HTTP; P_2 = REST; P_3 = SOAP
[b] D_1 = XML; D_2 = RSS; D_3 = ATOM; D_4 = JSON; D_5 = HTML; D_6 = CSV; D_7 = XLS; D_8 = RDF; D_9 = Image; D_{10} = Video
[c] DB_1 = Microsoft Access; DB_2 = DB2; DB_3 = MySQL; DB_4 = Oracle; DB_6 =PostreSQL
[d] These data sources are supported only if DAMIA is used in combination with Mashup Hub
[e] This is considered true if the source data has the same data model as the internal data model.

Murugesan [24] defines an API as an interface provided by an application that lets users interact with or respond to data or service requests from other programs, applications, or web sites. Thus, APIs facilitate the integration between several applications by allowing data retrieval and data exchange between applications. APIs help the developers access and consume resources without focusing on their internal organization. Simple and well-known examples of APIs include Open DataBase Connectivity (ODBC) and Java DataBase Connectivity (JDBC). On the Web, providers like Microsoft, Google, eBay, and Yahoo offer web APIs for retrieving content from their web sites. Such APIs generally use standard protocols such as REST/SOAP web services, AJAX (Asynchronous Javascript + XML) or XML RPC. APIs can also be used to access resources which are not URL addressable such as private or enterprise data [20]. However, some common data sources do not expose their contents through APIs. So, other techniques as screen scraping are needed to extract information.

3.3 Internal Data Model

As stated before, the objective of a Mashup application is to combine different resources, data in our case, to produce a new application. These resources come generally from different sources, are in different formats, and vehicle different semantics. To support this, each Mashup tool uses an internal data model. An internal data model is a single global schema that represents a unified view of the data [7]. A Mashup tool's internal data model can be either (i) *Graph-based* or (ii) *Object-based*.

In a *Graph-based model*, the graph refers to the model based on XML and those consumed as they are (i.e., XML). This can include pure XML, RDF, RSS, etc. Most of the Mashup applications use a Graph-based model as an internal data model. This is certainly motivated by the fact that most of

today's data available on the web are in this format and also, most of the Mashup tools are available via the Web. That is, all the data that are used by the Mashup tools, in this category, transform the input data into an XML representation before processing it. For example, *Damia* translates the data into tuples of sequences of XML data [5]. In an *Object-based model*, the internal data is in the form of objects (in the classical sense of the object-oriented programming). An object is an instance of a class which defines the features of an element, including the element's characteristics (its attributes, fields or properties) and the element's behaviors (methods). It should be noted that in this case, there is no explicit transformation, performed by the tool, like in the case of the graph-based model, but the programmer needs to define the structure of the object according to her data.

To illustrate the differences in the internal data models, let us consider the example of Figure 1 which shows an extract of a spreadsheet (or csv) file containing information about the national parks in the world[14].

title	link	description	pubDate
Grand Canyon National Park	www.grand.canyon.national -park.com /	Grand Canyon National Park...	19/03/2008
Big Bend National Park	http:// www.big.bend.national -park.com /	Big Bend National Park...	19/03/2008
Gulf Islands National Seashore	http:// www.hikercentral.com /parks/ guis /	Gulf Islands National Seashore.....	19/03/2008

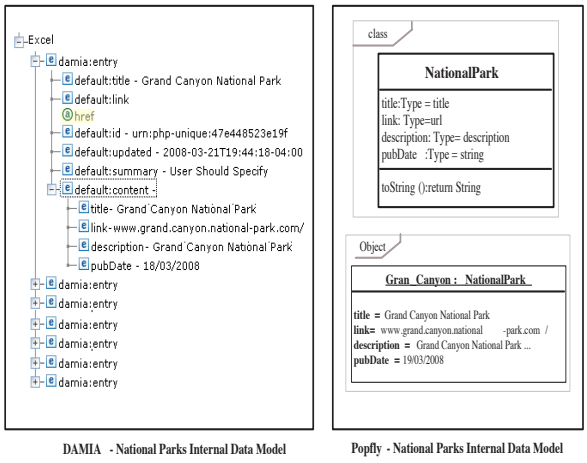
Figure 1: National Parks Data Source

The illustrated data in Figure 1 is given as an input to two different tools, namely *Damia* and *Popfly* and the obtained result is shown in Figure 2. Figure 2(a) shows the translation operated by *Damia* on the input data. That is, each row in the *csv* file is transformed to an XML representation contained in the element $< damia : entry >$. Each entry is composed of some elements containing general information regarding the data source such as file name (i.e., $< default : id >$), last updating (i.e., $< default : update >$) and by the $< default : content >$ element in which the national parks information is stored. Figure 2(b) shows how the data could be represented using an object-based notation is the case of *Popfly*.

3.4 Data Mapping

To instantiate an internal data model from an external data source, the Mashup tools must provide strategies to specify the correspondences between their internal data model and the desired data sources. This is achieved by means of data mapping. Data mapping is the process needed to identify the correspondences between the elements of the source data model and the internal data model [26]. Generally speaking, a data mapping can be: (i) *manual*, where all the correspondences between the internal data model and the source data model are manually specified, one by one, by the application designer. In this case, the tool should then provide some facilities for the user to design the transformation. (ii) *Semi-automatic*, where the system exploits some meta-data (e.g., fields names and types) to propose some possible mapping configurations. However, the user needs to confirm these alternatives and, usually, correct some of them. At this stage,

[14]Some elements are omitted for readability matters



DAMIA - National Parks Internal Data Model

Popfly - National Parks Internal Data Model

Figure 2: Representation of the National Park *csv* file in DAMIA and Popfly

only *Yahoo Pipe* supports the semi-automatic mapping, offering some hints for the user about possible mappings. (iii) *Automatic*, where all the correspondences between the two data models are automatically generated, without user intervention [26]. This is a challenging issue in the data integration area. Since the Mashup area is in its "early stage", this type of mapping is not supported by any Mashup tool. It should be noted that the mapping process may require an intermediary step, i.e., a wrapping step, in order to transform the source format to the internal format, e.g., from csv to XML. It is also interesting to point out that the mapping in the currently available Mashup tools is only done at schema level, while no semantic information is being considered so far. For instance, Semantic Web languages such as RDF[15] and OWL[16] can be exploited to manage the semantic part. However, this will need more investigation since it will introduce other complications due especially to the targeted users.

3.5 Data Flow Operators

Data flow operators allow performing operations either on the structure of the data (similar to the data definition language/operators in the relational model), or on the data (content) itself (similar to the data manipulation language/ operator in the relational model).

More concretely, data flow operators support: (i) restructuring of the schema of the incoming data, e.g., adding new elements, adding new attributes to elements; (ii) elaborating on a data set such as extracting a particular piece of information, combining specific elements that meet a given condition, change the value of some elements; (iii) building a new data set from other data sets such as merging, joining or aggregating data (similar to the concept of *views* in databases).

The implementation of the data flow operators depends strongly on the main objective of the tool, i.e., integration or visual-

[15]http://www.w3.org/RDF/

[16]http://www.w3.org/TR/owl-features/

ization. Some operators, e.g., *Union*, are implemented in different tools, e.g., *Damia* and *MashMaker*, but the attached interpretation is also different, e.g., a materialized union of two data sets in *Damia* and a virtual union of two Web pages in *MashMaker* (virtual in the sense that the schemas associated to the source pages are not altered with the new page). The main data integration oriented operators, implemented in several tools are the following: *Union, Join, Filter,* and *Sort*. We give hereafter a general description of these operators. A more detailed discussion of all the operators of all the considered tools is given in [22].

Table 2: Main and common operators

Operator	Description
Union	Combines two data sets in one containing all the data from the participating sets.
Join	Combines different data sets according to a condition.
Filter	Selects a specific subset (entities and attributes) from an original subset.
Sort	Presents the selected data in a specific order.

3.6 Data Refresh

In some cases, e.g., stock market, data is generated and updated continuously. Various strategic decisions, especially in enterprises, are generally taken according to the last status/values of the data. It is then important that a system propagates the updates of the data sources to the concerned user(s). There are two strategies dealing with the status of the data in the source, depending on the objective of the user: (i)*pull strategy* and (ii)*push strategy*[8]. The pull strategy is based on frequent and repetitive requests from the client. The pulling frequency is set to be lower than the average update frequency of the data in the source itself. The freshness of the data depends on the pulling frequency, i.e., the higher the pulling frequency, the fresher the data and vice-versa. One of the main disadvantages of a high refresh frequency is that unnecessary requests may be generated to the server. In the push strategy, the client does not send requests but needs to register to the server. The registration is necessary to specify/identify the data of interest. Consequently, the server broadcasts data to the client when a change occurs on the server side. The main disadvantage of this model is that the client can be busy performing other tasks when the information is sent which implies a delay in its processing.

Another important parameter to point out here is the way the tools manage the pull interval. We can define two possible strategies to handle this issue: a *global strategy* and a *local strategy*. For the *Global strategy*, the pull interval is set for the whole application. This supposes that the data sources have the same updating interval. That is, the data sources are requested at the same time interval, corresponding to the one of the Mashup tool. As a result, the user keeps better track of some sources (the ones having low refresh interval compared to the defined one) than the others (the ones having high refresh interval compared to the defined one). In the *Local strategy*, each data source is given its own refresh interval. This pull interval is supposed to correspond to the one of the data source refresh itself. As a result, a better trace is kept of each data source. From the tool's point of view, only *Damia* allows the users to define the pull interval and to handle it with the *Local strategy*. In fact, to set the pull interval, each source component has the *Refresh Interval* parameter. After the time has exceeded, the data from the specified URL is reloaded.

3.7 Mashup Output

We consider the output as a dimension in this study since a user might be interested in exporting her Mashup (the data flow) result in another format in order to reuse it or to process it with another particular application (e.g., spreadsheet) for further processing instead of visualizing it. That is, we can distinguish two main output categories: *Human oriented output* and *application oriented output*. In the *Human oriented output*, the output is targeted for human interpretation, e.g., a visualization on a map, on an HTML page, etc. That is, for this category, the output can be considered as the "final product" of the whole process. For the *processing oriented output*, the output is mainly targeted for machine processing. This is interesting in the case where the considered data needs to be further processed for, e.g., a knowledge discovery process. It should be noted that this category can, at some stage, include the first category, e.g., an RSS output can be at the same time visualized on an HTML page and also can be used by other applications for other processing tasks.

3.8 Extensibility

Extensibility defines the ability of the tool to support additional, generally user defined, functionalities. There can be two possible ways to define and use these functionalities. A functionality can be either (i) embedded inside the tool, i.e., the corresponding code of that functionality is added to the tool using a specific programming language, or (ii) external, i.e., invoking the corresponding service containing such function. Extensibility depends mainly on the architecture and the spirit of the tool. In some cases, the extension can be done by embedding the code of the desired functionality in the tool (e.g., *Popfly*); in other cases, services are invoked like REST services, SOAP, etc. (e.g., *Pipes*). In addition, this feature is managed differently by the different tools. In fact, in one case, the added function/service is shared with the whole community that uses the tool (e.g., *Popfly*). In the other case, the extension is visible only for the specific user (e.g., *Pipes*).

3.9 Sharing

Mashups are based on the emerging technologies of the Web 2.0 in which people can create, annotate, and share information in an easy way. Enabling security and privacy for information sharing in this huge network is certainly a big challenge. This task is made more difficult especially since the targeted public with the Web 2.0 is, or supposed to be, a general public and not expert in computing or security. This dimension defines the modality that the tool offers to enable resources sharing by guaranteeing privacy and security in the created Mashup applications. This is a challenging area in the current Mashup and a lot of work remains to be done. This dimension includes the following three indicators: 1) **What** is shared in the Mashup?, 2) **How** is this shared? and 3)**Who** are the users with whom this

(the shared resource(s)) is shared with? For the **What**, the shared resource can be *total*, *partial*, or *nothing*. The shared resource can be given different rights such as read only (user can read all entries but cannot write any entry), read/write (user can read and write all entries in the data), no access (user cannot read or write any entries). The **Who** as for it can be **All people**, **Group**, or **particular User**. Notice that for each member, different sharing policies (what and how) can be specified and applied.

For example, *GME* and *Yahoo Pipes* allow implementing sharing policies. In *GME*, the sharing policy can be: (i) total, i.e., read access to source code, data and output. (ii) Partial, i.e., read access to source code. (iii) Nothing, where the Mashup is not shared. When a Mashup is shared in GME, for the data used to build the application, the designer can decide to share it with a group or with all users by specifying Read/Write policies. In *Yahoo Pipes*, if a private element is used (Private string or Private text input) the code of the shared Mashup is available as well as the Mashup output but it is not possible to visualize the intermediate outputs.

4. DISCUSSION AND CONCLUSION

In this section, we make a general discussion on the tools by considering their advantages and disadvantages. We aim, at the same time, to give possible points to consider for further improvements.

Mashup tools are mainly designed to handle Web data. This can be seen as an advantage, but an inconvenience at the same time. In fact, it is an advantage since it offers access and management of some data available only on the Web, e.g., RSS feeds. To access these Web data, the tools support the two most used protocols for exposing APIs, i.e., REST and SOAP protocol. This is a consequence of the success, utility and the popularity of these protocols. A disadvantage is that data available on desktops can not be accessed and used the same way. This is a considerable disadvantage since users bring a lot of data onto their desktops for cleaning, manipulation, etc. There is a lot of work done to help the user put and manage data on the web [16], but since this is not completely adopted, supporting local data on a user's desktop should be considered.

The majority of the tools have an internal data model based on XML. This design choice is motivated by the fact that the data available on the web is mainly exposed in an XML format. Also, the communication protocols for the data exchanging over the network use generally XML messages. The other dominant internal data model in Mashup tools is object based. This data model is much more flexible to use, even if more programming is required to implement operations on it especially for programmers. This diversity can explain their targeted/origin community. In fact, XML is much more for databases community where as object is for applications and development community.

To manage data, the tools make available only a small set of operators for data integration and manipulation. The set of provided operators is usually designed based on the main goal of the tool. For example, if the tool is visualization oriented, only few operators for data elaboration such as filtering and sorting are available. In addition, the offered operators are not easy to use, at least from a naive user point of view. Also, the tools do not offer powerful expressiveness since they allow expressing only simple operations, e.g., simple joins, and can't be used to express more complicated queries such as joins with conditions, division, etc. This means that, from the expressiveness point of view, these tools are far from reaching the database languages, i.e., integration languages, such as SQL.

None of the analyzed tools implement a *Push strategy* for the data refreshing and the reason is that the majority of the currently available APIs are REST based. The style of the REST protocol requires all communications between the browser and the server to be initiated by the client and no support is offered to maintain the state of the connection [9]. All the analyzed tools use a *Pull strategy* for data freshness handling. This can be motivated also by the fact that the tools providers wish to control (or prevent) the overloading of their servers. In addition, they implement a *Global strategy* for the pull interval setting. This strategy however does not allow developing applications in which processed data are characterized by a high refresh frequency, since it is not possible to explicitly specify the refresh rate for each source.

One of the main goals of Web 2.0 technologies is the creation, the reuse, the annotation and the sharing of web resources in an easy way. Based on these ideas, Mashup tools are all extensible in the sense that new operators, and in some cases data schemas, can be developed and invoked or/and plugged inside the tools. However, at this stage, the majority of tools do not support the reuse of the created Mashups. This feature could allow developing complex applications by integrating the results of different Mashups (also built with different Mashup tools). Some tools start to consider this issue such as Potluck[19][17] which can use the Exhibit output. However, this is a limited cooperation between tools. This is a very important point especially that the tools have a several limitations and a user can't express his wishes using only one tool.

The current development of Mashup tools is mainly focused on offering features to access, manage, and present data. Less consideration has instead been given to the issue of data sharing and security so far. The security criterion needs to be taken into account inside the tools since communication problems could make a Mashup perform too many requests to source data servers, causing overloads for those servers. At this time, only *Intel Mashmaker* takes into account this problem applying some performance restrictions on the Mashup application[14].

Also, all the analyzed tools are server side applications, meaning that both the created Mashup and the data involved in it are hosted on a server owned by the tool provider. Therefore, the tool provider has the total control on the Mashup and if a user wants to build an application containing that Mashup, the dependability attributes of that application cannot be properly evaluated. In addition, from the performance point of view, no tools provide information regarding the analysis of the performances and, in particu-

[17]This tool is not discussed in this paper for some of the reasons introduced in the beginning of this paper.

lar, information regarding the evaluation of scalability. That information is needed to know the capability of a system to handle a growing amount of data and the user requests.

Finally, all the tools are supposed to target "non-expert" users, but a programming knowledge is usually required. Some tools require considerable programming effort since the whole process needs to be implemented manually using instructions expressed in programming language such as Java Script. Others necessitate medium programming effort given that only some functionalities need to be coded in an explicit way using a programming language; a graphical interface is offered to the user to express most of operations. At this time, there is no tool that requires low or no programming effort by the user to build a Mashup, which is necessary to claim that the tools are targeted for end-users.

5. REFERENCES

[1] *Mashup Styles, Part 1: Server-Side Mashups, http://java.sun.com/ developer/ technicalArticles/J2EE/mashup_1/.*

[2] *Mashup Styles, Part 2: Client-Side Mashups, http://java.sun.com/ developer/ technicalArticles/J2EE/mashup_2/.*

[3] *OASIS: Web Services Business Process Execution Language Version 2.0. (2007), http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html.*

[4] S. Abiteboul, O. Greenshpan, and T. Milo. Modeling the mashup space. In *WIDM*, pages 87–94, 2008.

[5] M. Altinel, P. Brown, S. Cline, R. Kartha, E. Louie, V. Markl, L. Mau, Y.-H. Ng, D. Simmen, and A. Singh. Damia: a data mashup fabric for intranet applications. In *VLDB '07*, pages 1370–1373. VLDB Endowment, 2007.

[6] S. Amer-Yahia and A. Y. Halevy. What does web 2.0 have to do with databases? In *VLDB*, page 1443, 2007.

[7] C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Comput. Surv.*, 18(4):323–364, 1986.

[8] M. Bhide, P. Deolasee, A. Katkar, A. Panchbudhe, K. Ramamritham, and P. Shenoy. Adaptive push-pull: Disseminating dynamic web data. *IEEE Transactions on Computers*, 51(6):652–668, 2002.

[9] E. Bozdag, A. Mesbah, and A. van Deursen. A comparison of push and pull techniques for ajax. In S. uang and M. D. Penta, editors, *Proceedings of the 9th IEEE WSE*, pages 15–22, 2007.

[10] F. Curbera, M. J. Duftler, R. Khalaf, and D. Lovell. Bite: Workflow composition for the web. In *ICSOC*, pages 94–106, 2007.

[11] F. Daniel, J. Yu, B. Benatallah, F. Casati, M. Matera, and R. Saint-Paul. Understanding ui integration: A survey of problems, technologies, and opportunities. *IEEE Internet Computing*, 11(3):59–66, 2007.

[12] C. Duda, G. Frey, D. Kossmann, and C. Zhou. Ajaxsearch: crawling, indexing and searching web 2.0 applications. *PVLDB*, 1(2):1440–1443, 2008.

[13] S. Dustdar and W. Schreiner. A survey on web services composition. *International Journal of Web and Grid Services*, 1(1):1–30, August 2005.

[14] R. Ennals and M. N. Garofalakis. Mashmaker: mashups for the masses. In *SIGMOD*, pages 1116–1118, 2007.

[15] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[16] R. Geambasu, C. Cheung, A. Moshchuk, S. D. Gribble, and H. M. Levy. Organizing and sharing distributed personal web-service data. In *WWW*, pages 755–764, 2008.

[17] A. Halevy. Why your data won't mix. *Queue*, 3(8):50–58, 2005.

[18] D. F. Huynh, D. R. Karger, and R. C. Miller. Exhibit: lightweight structured data publishing. In *WWW '07*, pages 737–746, New York, NY, USA, 2007. ACM.

[19] D. F. Huynh, R. C. Miller, and D. R. Karger. Potluck: Data mash-up tool for casual users. In *ISWC/ASWC*, pages 239–252, 2007.

[20] A. Jhingran. Enterprise information mashups: integrating information, simply. In *VLDB '06*, pages 3–4. VLDB Endowment, 2006.

[21] S. Kinsella, A. Budura, G. Skobeltsyn, S. Michel, J. G. Breslin, and K. Aberer. From web 1.0 to web 2.0 and back -: how did your grandma use to tag? In *WIDM*, pages 79–86, 2008.

[22] G. D. Lorenzo, H. Hacid, H. young Paik, and B. Benatallah. Mashups for data integration: An analysis. Technical Report UNSW-CSE-TR-0810, 2008.

[23] E. M. Maximilien, H. Wilkinson, N. Desai, and S. Tai. A domain-specific language for web apis and services mashups. In *ICSOC '07*, pages 13–26, Berlin, Heidelberg, 2007. Springer-Verlag.

[24] S. Murugesan. Understanding web 2.0. *IT Professional*, 9(4):34–41, July-Aug. 2007.

[25] W. Pree. *Design patterns for object-oriented software development*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1995.

[26] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001.

[27] D. E. Simmen, M. Altinel, V. Markl, S. Padmanabhan, and A. Singh. Damia: data mashups for intranet applications. In *SIGMOD '08*, pages 1171–1182, New York, NY, USA, 2008. ACM.

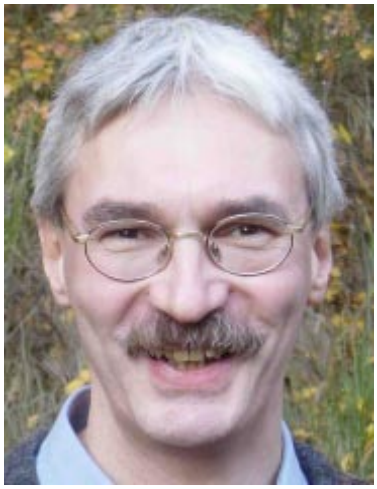
[28] J. Wong and J. Hong. Marmite: end-user programming for the web. In *CHI '06*, pages 1541–1546, New York, NY, USA, 2006. ACM.

[29] J. Yu, B. Benatallah, R. Saint-Paul, F. Casati, F. Daniel, and M. Matera. A framework for rapid integration of presentation components. In *WWW '07*, pages 923–932, New York, NY, USA, 2007. ACM.

Gerhard Weikum Speaks Out

on Why We Should Go for the Grand Challenges, Why SQL Is Too Powerful, the Myth of Precision, How to Have a Big Research Group in Germany, and More

by Marianne Winslett



Gerhard Weikum
http://www.mpi-inf.mpg.de/~weikum/

Welcome to this installment of ACM SIGMOD Record's series of interviews with distinguished members of the database community. I'm Marianne Winslett, and today[1] *we are at the VLDB 2007 conference in Vienna. I have here with me Gerhard Weikum, who is a Research Director at the Max-Planck Institute for Informatics in Saarbrücken, Germany. Gerhard is an ACM Fellow and the president of the VLDB Endowment. He has a SIGMOD Best Paper Award, a VLDB 10-Year paper award, and a CIDR Timeless Idea Award. His PhD is from the University of Darmstadt. So, Gerhard, welcome!*

Most of our readers and viewers won't know that a German university has at most one professor in an area. So, by definition, you are the only *database professor at the Max Planck Institute for Informatics. From a US perspective, this guarantees that a CS department doesn't have critical mass in any one area. How have you built such a strong group under that system?*

The main thing to watch out for is having good people around you. Great students are a wonderful asset. If you have a strong team of great students, you can do good work with few people.

Let me also correct a little bit your perception of the German system. In Germany, you become a professor only at a relatively mature stage, and then you would traditionally have tenure right away. Only gradually are we adopting elements from the Anglo-American system, where you become a non-tenured assistant professor earlier. Obviously you cannot be a graduate student

[1] The print version of this interview has been revised slightly, to bring it up to date.

and then disappear into the world for ten years, and then come back as a professor. So during that intermediate period, people have titles such as *post-doc*, *research associate*, and so on. I have a few of these people as well.

I am at the Max-Planck Institute, but I am affiliated also with the University of Saarbrücken, which is on the same campus. So I have a colleague at the university as well. This used to be Christoph Koch; since fall 2008 it is Jens Dittrich, who came to Saarbrücken from ETH Zürich – the same path, by the way, that I took 15 years ago. Jens is a great colleague, and we have started collaborating.

You work in many different areas now, but you started out narrowly focused on transaction processing. What led you to switch from a narrow focus to such a broad one?

I am curiosity driven. I want to learn new things, understand new things, and the best way of doing this is by working on a new area. It forces you to understand the state of the art, and dig into the existing literature. Typically I start by teaching the subject or the field. For example, when I moved towards information retrieval seven years ago, I started by teaching an information retrieval class. By teaching the class, you also get research ideas.

When I look back, I think my career roughly consists of three parts. First was the transaction processing. (By the way, at that time we did not think of transaction processing as being *narrow*!) Then in the early nineties, I started working on the theme of automatic tuning, and that went on for about ten years. Then around 2000, I moved to the field of database and information retrieval integration.

You don't think you have gotten any broader over the years?

I have gotten broader, because there is some temporal overlap between these three areas. I do not completely stop working on all aspects of the previous theme, once I move on to a new theme. But some of the overlap is not exactly original research. For example, on the automated tuning theme, recently I have been doing tutorials together with Surajit Chaudhuri. That is not original research. Surajit is still doing original research; I am involved, essentially, in the educational part.

I have more students than I used to, because at the Max-Planck Institute, you are supposed to build up a big group, so that you have critical mass. At the university, before I joined the Max-Planck Institute, I worked with a handful of students. I think I was more focused. Now we have 20 people in the group, of which 10-15 are students and the rest are post-docs or people who, in the US, might be considered as young assistant professors. I tend to give some leeway to talented students. If they come with their own creative idea, and if they convince me that it is a good direction, and they have the talent to tackle the issues, then I let them go. So some of our work (such as what we did on machine learning) is prompted by a student having a good idea, and then I join as an advisor and get involved.

How do you keep yourself from being spread too thin?

That is a good point. You have to continuously monitor yourself and see if you are focused. I try to have one or two main projects, typically centered around some kind of systems-building activity. But we don't build the system just to do systems-building work; the system is the focal point of the research activity, and an integration point for all the sub-projects. How to get students to talk to each other is a great worry, because they work from different angles on the

same system and need to understand how their components interact. When I was at the university working with 5-10 people, we typically had one system-oriented project plus something more half-baked and adventurous that could lead to a larger activity in the future. Now, at the Max-Planck Institute, we have two or three of these system-centric projects, just because of the size of the group. I have two roles to play: a research manager in all of the projects, and a researcher in a subset of them.

You have moved from an area where the work is very precise, to one where everything is very fuzzy. What led to that transformation?

I think that the notion of precision is a myth! In the good old days, when you were working on payroll data, maybe things really were precise. My salary, hopefully, is precise. But when you talk about scientific data such as measurements, when an experiment gives you a value, there is always an error bar, some degree of uncertainty. We just ignore that uncertainty. We pretend that yes, the temperature was exactly 35.5789 degrees Celsius, but that is not the case. In the text world, or in integration of text and data, there is inherent uncertainty in interpreting the text and sometimes also in the information need on the user's side. For IR people, a query is an approximation to the user's information demand. It is not a dogmatically given precise query.

It sounds like you are saying that you were ready to face reality.

Definitely one cannot be precise in a world of text data, such as when extracting entities and relationships from text. It is important to capture the confidence in what you are doing. As you move on to aggregating the data, composing the data, and querying the data, eventually you may get very far off from your original starting point, and that might lead to misinterpretation of what the query results mean. So it is important to capture the confidence, or lack of confidence, at each point along the way and understand how that confidence propagates to later processing steps. I am a deep believer in probabilistic data.

Working in many areas puts you in a good position to tell us where the database field is going, and what some of the interesting problems are for the future.

This is highly subjective, obviously, so my answer will match what my own group is working on. I see an enormous explosion of information everywhere. To me, text is one of the kinds of information that humans produce most efficiently. Precise data, that is, schematically structured data, just takes too long to produce. At its central core, a whole essay might not say much more than one tuple in a database, but most people can produce the essay more easily than the tuple describing the gist of the essay. So I expect to see a continuing explosion in text data, connected with fuzzy structures such as Web 2.0, blogs, and bulletin boards, with tagging, bookmarking, and all those kinds of things.

People can produce speech even more efficiently than text, and speech recognition is getting much better. I see potentially a big wave of speech processing that can extract interesting elements from the speech itself, such as the speaker getting very emotional, getting very upset, being ironic, and so on.

Yes, but you cannot tell me that speech recognition is database research.

I don't care which drawer you put me in: I am a computer scientist. I think it is good to look across the fence at other areas. I am not saying speech recognition is a database issue; it has a data management component, and lots of other components.

Someone suggested that in light of your move from things that are precise to things that are statistical, you may be being "seduced by the dark side" and end up doing AI. Do you think that you are heading in that direction?

That would be perfectly fine, I have nothing against this. When I was young, I also thought in terms of idioms like "being seduced by the dark side." Mike Stonebraker used to say, "This problem is AI-complete," which meant that the problem was beyond any hope of solution, was science fiction, was on the same level as "Scotty, beam me up!" Now, as I grow older, I think this attitude is wrong.

We should go for the grand challenges. Physicists also go for grand challenges, such as controlled fusion, and they have huge projects in areas such as plasma physics. And no, they don't keep their promises. They say that they will have a breakthrough in thirty years, and it doesn't happen. That tells us that their problem is hard. In the life sciences, in understanding not just the syntax of the human genome but also its semantics, its functionality – this is a century-long project. So, I think we computer scientists need to grow up and become more self-confident, and go for the really grand challenges. Then we can break the challenge down into steps; obviously it does not make sense if we spend 30 to 50 years with no visible progress. If we gave ourselves a goal whose validation or falsification is a hundred years in the future, we would not live long enough to see whether we are wrong or we achieved a breakthrough. So we need to break our grand challenge down into milestones, and into short- and intermediate-term goals as well. The grand challenges are out there, and we should pursue them.

Tell us about your 2002 VLDB 10-year paper award.

That refers to a paper we published in 1992 in the VLDB in Vancouver. Our paper was on load control for lock management. You can run into lock contention situations under extremely high load, like load surges with mixed workloads. Our paper was about how to do admission control and concurrency control, so it was very narrow and very technical. I don't think we got the 10 year award for the paper itself, but for the theme behind it: automatic tuning.

In 1989 I came back from my post-doc time at MCC in Austin, and took a position as an assistant professor at ETH Zurich. At MCC, we had gone after a grand challenge: a massively parallel database machine, which was a big thing back then. I was influenced by that project, so I wanted to work on something really challenging and long term, but at the same time something off the beaten path that not too many other people would work on. I was a young assistant professor, and I didn't have so many resources anyway.

I came up with the theme of trying to automate tuning by putting more smartness in the system. We pursued adaptive methods, we pursued stochastic models, and we pursued feedback control, which was the major element behind this VLDB 1992 paper. I think that when the committee picked us for the 10-year award in 2002, they were acknowledging the importance of the theme that we had started working on: automatic tuning, self-managing systems, or autonomic computing, as it is called now.

Tell us about your 2005 Timeless Idea award from the Conference for Innovative Database Research (CIDR).

That is kind of a fun award; it is very different from the other awards I have received. I have listed it as well as the more serious awards on my web page, because I think people should not

overrate the serious awards. There is some luck involved in receiving one of them, and so there are many good people who deserve one and have not yet gotten one.

The CIDR award refers to a late-night event called the Gong Show that takes place at the conference. Twenty to 25 people each get five minutes to talk about whatever they want: challenges, unsolvable problems, their pet problems, jokes, whatever. People are in a very loose mood at 10 in the evening, after some drinks, and my presentation was semi-serious. My theme was the experimental methodology in our field. Obviously this is a serious theme, but the presentation itself was more on the funny side. I showed some tricks about how you can massage your performance charts and make them look better than they are. My real point was that we should carefully reflect on the experimental methodology in our field. We are really kids in this regard, compared to other fields. The slides for my presentation are available on the web, by the way, and the last one or two slides have some serious thoughts on what to do about this problem. This was a very timely topic, considering the recent change at SIGMOD to include experiment repeatability for accepted papers.

What do you think of SQL?

As a language? I have never worked on designing a language or on other language issues, but I have considerable teaching experience. When I teach SQL, I want to give formal semantics, like translations to relational algebra or to relational calculus. You can do this easily for a narrow subset of SQL, like select, project, join. But once you have complex SQL queries with five levels of nested subqueries, a handful of left outer joins, correlated variables, and aggregations and groupings, hardly any students get such queries right. We teach these features by example: the teacher gives one example of how to use a new feature and hopes that the students will magically know how to use the feature correctly. But the students don't.

I think that the result of this teaching methodology is that our students will produce lousy SQL code later on, with bugs that must be found and removed. It is not easy to debug SQL, because it is so declarative. The more declarative a language is, the harder it is to debug. Debugging SQL typically involves breaking a complex statement down into simpler SQL building blocks. But then, why do you want such a super powerful language?

So I think there is virtue in having a language that does less. My role model for this is people like Niklaus Wirth, the inventor of Pascal. His languages have a *less is more* philosophy. They have the right, easy to use building blocks. The building blocks are composable, and the language doesn't have too many features. Feature richness is a curse. I know this is debatable, and many people would disagree. This is just my opinion, my two cents, nothing else.

If the more declarative a language is, the harder it is to debug, then assembly code should be the easiest to debug.

That is a good comment! In terms of state-driven debugging, where you look at the state of a program, that is actually true, or close to true. But ease of debugging is not the only important aspect of a language. The amount of time required to design, code, and reason about a program is also important.

I am not saying declarative programming is wrong. I am just saying that if you go all the way to being declarative, then you also have to be extremely precise in terms of the language's semantics, and you have teach this semantics. Hoping that the students understand the super-complex semantics from a few examples cannot work.

There is a TODS paper that gives a formal semantics for SQL.

It is tough to teach from that paper. And by the way, this is not specific to SQL, this is about all declarative languages. I think making them too rich is an issue. There is no silver bullet here.

You are a director at a Max-Planck Institute, and you are the president of the VLDB Endowment. Do you have further organizational ambitions?

Oh, definitely not! I am taking on these kinds of service positions because I think someone has to do them, and I feel indebted to the community. When I was young, I attended conferences and enjoyed them. Someone has to organize them. And someone has to organize the organizations behind the conferences. So I think I am just doing my duty.

I have been told that you have great technical, administrative, and meeting skills. Since people like that are in high demand, how do you balance your time between service to the community, managerial roles, and research?

I don't think I am particularly better at these things than other people. When I accept a job, I try to do it right. That is my attitude; it is probably in my genes. But I am certainly not a natural born manager or chair of anything. When you say I have great skills in these kinds of things, this is just because I put time and work into this.

I am trying to limit the time I put into these kinds of service activities. I am still at a phase where I feel indebted to do these things. At some point, I will think that I have paid back my debt, and then I will refuse them again. At that point maybe I will write a book again, or something of that kind.

What killed that nice Parallel and Distributed Information Systems (PDIS) conference?

Oh, that was Jeff Naughton! Jeff and I were the PC co-chairs of the last PDIS conference, held in Miami in 1994. Actually, I am just kidding. Jeff and I killed PDIS together! The truth is that PDIS killed itself automatically, and there is nothing wrong with that.

There are specialized conferences such as PDIS, Deductive and Object Oriented Database Systems (DOODS), and some of the workshops that repeat year after year. These specialized conferences and workshops make sense as long as their topics are not sufficiently well covered by mainstream conferences like VLDB, SIGMOD, etc. But when they run out of steam, either in terms of submissions or attendance, because the mainstream conferences have absorbed these directions, then there is nothing wrong with letting them phase out. So PDIS just phased out.

I see what you are saying, although since there is such a submissions crunch at the main conferences, it seems like we should be able to have some good specialized conferences.

Things are different now. For example, the web space is exploding, so there might be a case for a Web 2.0 conference. But of course there are already new conferences in that space. Unlike the typical DB guy, I am also observing what's going on in our neighbor communities, IR and web research. For example, there is a new conference called WSDM, pronounced "Wisdom", which is for all search- and mining-related issues on the web, including Web 2.0.

I hear that you like to take solo vacations that involve crawling on your belly in the desert. Is that how you would describe them?

I like the desert. I have spent significant time in the US, and I like the Southwest. Yes, I have done some backpacking solo trips. The crawling is not by design, but sometimes you end up in a space where you cannot backtrack. I did some canyoneering, for example, in the Southwest, in southern Utah. Sometimes you are at an obstacle, and you can get forward, but you cannot get backward, because you already went down some obstacles, like steep drops over boulders and small waterfalls and that kind of thing. And sometimes you have to crawl underneath boulders.

If it is that dangerous, is it wise to be out by yourself?

It is no more dangerous than other things. I have had in my life a few injuries; one of them was a torn ligament in the ankle. That happened while I was waiting for someone on the sidewalk! So, hiking by myself is no more dangerous than other things.

True, although when you tore your ligament on the sidewalk, there were people all around you. If you are off in the desert in an obscure spot, no one will be around you. And your cell phone might not work either.

That is true. In fact, I don't carry a cell phone on these trips, because it is very clear that it won't work. Typically you register with some agency, the Bureau of Land Management or a national park. A few days after you were supposed to return, they would start searching for you. And I am a careful guy during these trips. I don't take unnecessary risks.

What do you think of the European funding agencies' emphasis on very large projects?

"Large" can mean different things; you were probably referring to the number of partners in the project. There are different kinds of large projects. Those with 20 partners, or maybe even more, are really *networks*, so they break down into subprojects with perhaps five partners. There is some loose coupling between the subprojects, in the sense that people talk to each other, but there are no joint development goals across subprojects. So these kinds of large projects are fine. The many-partner projects I have been involved with have been networks.

Some many-partner projects are designed to have hard deliverables in the form of systems at the end of the project. If you strictly evaluated such projects against their officially declared goals, then they would be bound to fail. I do not believe that 20 partners can do joint development, especially if half of them are academic partners, and no partner has a big budget.

There are also what are called STREPs – Specifically Targeted REsearch Projects, perhaps – I am not sure what the acronym stands for. These typically have 5-10 partners in different roles. There might be 3-4 technology partners and research partners, and the other partners would be technology transfer or requirements analysis partners. Sometimes, when companies are involved in these projects, they can play different roles. They could play a major development role and a technology role; but they could also play a role of supplying use cases, requirements, or potential business models a few years down the road. The latter type of partner is involved in the early and late phases of these projects. That reduces the remaining partners to just a handful, who are the core technology and development and research partners. That model works.

Do you have any words of advice for fledgling or midcareer researchers or practitioners?

I know quite a few people in computing theory, where there are so many young superstars, and then there are the 40 year olds. In Europe, if you are a decent theoretician, but you are not among the stars and you are approaching 40 and you have not gotten a tenured professor position, you might be in trouble, because there are always these 25 year old new superstars. These 40 year olds are not bad researchers. They have good skills, so they might consider moving to another area where they can leverage their expertise and their skills and methodology. The irony is that I think the demand for researchers on the practical side is much higher, but there is a disproportional fraction of talent going into theory because it looks more challenging. So there are people who are just not good enough to be a superstar in theory, but could do very well by moving to a more practical field.

From among your past papers, do you have a favorite piece of work?

There are a few I like very much and I am very proud of. One that is lesser known is a workshop paper in WebDB 2000. We started working on ranked retrieval for XML, and we had this little paper, just 6 pages, for which we coined the title "Adding relevance to XML." At that time, the XML wave was about to take off, but it was all schema driven. I thought there was no way of unifying the world into the "right" schemas. If you have heterogeneity, diversity, and ambiguity, then you need ranking. This was our first paper on DB and IR integration.

If you magically had enough time to do one additional thing at work that you are not doing now, what would it be?

Write a book with Surajit Chaudhuri.

On automatic tuning?

You got it!

If you could change one thing about yourself as a computer scientist, what would it be?

I should have learned more theory when I was young – more math and more theory. I see that this can be leveraged. You can be much more effective and efficient at certain things when you have the right base ingredients.

Thank you very much for talking with me today.

Thank you, Marianne.

Open Forum - Call for Opinions

The Role of Workshops in the Dissemination of Research Results

The Open Forum Column was introduced with the March 2008 issue as a forum for members of the broader data management community to present (meta-)ideas about non-technical issues and challenges of interest to the entire community. With this issue, we are taking the next (evolutionary) step for the column, namely opening the floor to comments and opinions from the entire community on a *specific topic*. The goal is simple: allow more members of the community to express their opinion about a particular topic and hopefully identify a consensus, whenever possible.

The first topic will be the "Role of Workshops in the Dissemination of Research Results". In the past, it used to be that workshops were a venue for immediate feedback on new, not fully-developed ideas (and typically did not have official proceedings). Works presented in workshops typically materialized subsequently into full-fledged conference papers, which in turn were later published in an archival journal (with an addition of about 30% of new material from the conference version).

This system seems somewhat broken right now, as conferences seem to be adopting a strict double submission policy which considers duplicate the submission of a paper with an earlier workshop version of over four pages (most workshops have 6+ pages), even if the workshop did not have official proceedings. On the contrary, short papers accepted at conferences do not violate this rule, despite the significant overlap and the fact that most probably, virtually the same submission is generating two papers (a short paper at the first conference and a regular paper at the second conference).

Related to this question is, of course, the broader question of what kinds of topics are appropriate for workshops, whether it is good to have workshops running multiple years or have new workshops every year, etc.

If you have an opinion on the topic, I invite you to *share it with the data management community at large*, by submitting it to the Open Forum Column. The logistics are as follows:

- We will devote two issues to this topic:
  - June 2009 (**opinions due by July 5th**) and
  - September 2009 (**opinions due by August 1st**).
- To submit your opinion, you need an account on RECESS, the SIGMOD RECord Electronic Submission System, at `http://db.cs.pitt.edu/recess` You need to use your "work" email address to register; this email address will not be used for anything other than correspondence for your paper(s).
- Please submit your opinion under the "*Open Forum: the Role of Workshops*" column.
- By default, opinions will be published with attribution to the author. If you want your opinion to be published anonymously, please indicate it in the first line of the text.
- Your opinion is to be formatted in **plain text** and be less than 300 words. We will not accept doc/pdf submissions, given the complexity of aggregating them into a single document.
- The goal (and hope) is for all opinions to be published without editing. To achieve this, you must refrain from personal attacks (e.g., Dr. BadPCchair rejected my XYZ 2007 submission unfairly). Reporting on personal experiences is ok, if it is constructively used to support an argument, but please refer to the workshop/conference/journal without a year qualification, if possible. Finally, reporting on how other research communities are dealing with this issue is also acceptable.

Alexandros Labrinidis
May 2009

The Computing Innovation Fellows Project

An Opportunity for New PhDs in Computing-related Disciplines

Project Description. The Computing Innovation Fellows (CIFellows) Project (see http://cifellows.org) anticipates making up to 60 grants to new PhD graduates in Computer Science, Computer Engineering, Information Science, and closely related fields. Grants will provide funding for one year of work with a mentor at a host organization. Awardees (henceforth "CIFellows") will be eligible to apply for a second year of funding to follow the first year.

The project seeks to involve many organizations and sub-disciplines in order to encourage cross-flow and broad participation. In particular, it is anticipated that no more than two CIFellows will be awarded to graduates of the same institution in 2009 and that no host organization will host more than two CIFellows. It is expected that the project will involve approximately 50 different organizations.

Each CIFellow must have a mentor at a host organization. CIFellows may engage in research, teaching, and/or other activities, such as public policy work, that contribute to the advancement and vitality of computer science and allied disciplines.

Eligibility. A CIFellow must have completed all requirements for graduation from a U.S. PhD program between May 1, 2008 and August 31, 2009. The PhD must be in Computer Science, Computer Engineering, Information Science or a closely related field. Preference will be given to U.S. citizens and permanent residents, but others will be considered.

Award Size and Duration. Awards will be for $75,000 salary for 12 months with approximately $25,000 for fringe benefits and a $15,000 allowance for moving, travel, and discretionary expenses. Host organizations will receive indirect costs at the 25% rate. The 12-month assignment must begin by November 1, 2009.

CIFellows Website. The CIFellows website can be found at http://cifellows.org. This website contains a number of resources including a FAQ, application materials and submission instructions, information for mentors, and an area for prospective mentors to indicate their interests.

Proposal Content and Format. A complete application comprises the following materials submitted by June 9, 2009:

- A one page statement of research accomplishments
- A two-page statement of goals for scholarly activity, teaching, and/or other proposed activities including a statement of the intellectual merits and broader impacts of the proposed work
- A curriculum vitae
- Two confidential letters of reference submitted directly by the recommenders following the instructions on the website; one of these letters will normally be from the applicant's PhD advisor
- A letter from the applicant's PhD advisor or department chair stating that the applicant will have completed all work toward a PhD degree by August 31, 2009
- A list of one to three proposed mentors who must be from U.S. institutions different from the applicant's graduate institution. Note that the goals of the CIFellows project favor selecting mentors at a broad range of institutions.

- For each proposed mentor:
    - A one page document from the applicant indicating the sub-discipline of the proposed work with this mentor (from the CIFellows website – http://cifellows.org), the rationale for this choice of mentor and the expected benefits of working with this mentor, as well as the relationship of the activity to the applicant's career goals.
    - A one page document from the proposed mentor indicating a commitment to work with the prospective CIFellow, a mentoring plan, and a summary of the mentor's prior record of mentoring (see "Information for Prospective Mentors and Host Organizations" below).

Identifying Mentors and Mentees. The CIFellows website provides a facility to help applicants identify mentors and to allow prospective mentors to express interest in mentoring a CIFellow. Postings are organized by sub-disciplinary categories and mentors may place short postings on this site under one or more of these categories. Mentors are encouraged to include their mentoring credentials and planned mentoring activities on this site along with contact information to allow prospective CIFellows to contact them. Applicants are not required to choose mentors from this website.

Information for Prospective Mentors and Host Institutions. Once a prospective mentor has agreed to support an applicant's candidacy, the mentor must provide the applicant with a one-page letter of support including the mentoring plan. In addition, the mentor must indicate that the mentor and the host organization are committed to hosting the CIFellow. This document comprises part of the candidate's application materials (See "Proposal Content and Format" above).

Evaluation of Proposals. Proposals will be evaluated by a broadly representative committee from industry and academia. Applications will be evaluated based on:

- The applicant's record of research and other accomplishments as they relate to the proposed work
- The strength of the proposed activity with respect to intellectual merit and broader impact
- The quality and suitability of the mentoring plan for the CIFellow's future work
- Contribution to the project objectives of enhancing cross-flow between institutions and diversity of represented sub-disciplines

Since it is anticipated that at most two fellows will be selected from any institution and at most two CIFellows will be hosted by any organization, applicants are encouraged to identify up to three possible mentors at different organizations. If the application is successful, the award will be for work with a specific mentor. Applications from women and underrepresented minorities are strongly encouraged.

Awards are expected to be announced by July 10, 2009.

Award Administration. Awards will be administered through the Sponsored Projects Office at the host institution. CIFellows and their mentors will be required to complete short reports near the end of each funded year.

For Further Information, please consult http://cifellows.org. Additional questions may be directed to contact@cifellows.org.

Call for Participation: **ACM SIGMOD/PODS 2009 Conference**

Providence, Rhode Island
June 29 - July 2, 2009
`http://www.sigmod09.org`

2009 ACM SIGMOD International Conference on Management of Data
2009 ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems

For decades, the joint ACM SIGMOD/PODS Conference has established itself as the top data management conference in the world for researchers, practitioners, developers, and users to report and share cutting-edge ideas and results, and to exchange techniques, tools, and experiences. We are delighted to invite you to attend ACM SIGMOD/PODS, to be held in Providence, Rhode Island, from June 29 to July 2, 2009.

The conference will take place at the Westin Providence Hotel, where a block of rooms has been reserved for attendees. Rooms can be booked at a special conference rate, including a special rate exclusively for students, either online from the conference web site or over phone by quoting "SIGMOD/PODS".

****** **Registration is now open: http://www.sigmod09.org/** ******

The deadline for early registration is May 29, 2009. The hotel's room block will be released after May 30, 2008, so you should make your reservations well before then to avail of the special conference rate. The conference banquet will be held in historic Newport, a popular New England summer resort renowned for its mansions.

The highlights of the SIGMOD/PODS technical program are as follows:

SIGMOD

2 Keynote presentations

- *Enterprise Applications - OLTP and OLAP - Share One Database Architecture*. Hasso Plattner (Hasso-Plattner-Institute for IT Systems Engineering)
- *Transforming Data Access Through Public Visualization*. Fernanda B. Viegas (IBM), Martin Wattenberg (IBM)

5 Tutorials

- *Large Scale Uncertainty Management Systems: Learning and Exploiting*. Your Data. Shivnath Babu, Sudipto Guha, Kamesh Munagala
- *FPGA: What s In It For a Database*? Rene Mueller, Jens Teubner
- *Keyword Search on Structured and Semi-Structured Data*. Yi Chen, Wei Wang, Ziyang Liu, Xuemin Lin
- *Database Research in Computer Games*. Alan Demers, Johannes Gehrke, Walker White
- *Anonymized Data: Generation, Models, Usage*. Graham Cormode, Divesh Srivastava

Panel on the 40th anniversary of relational data model
63 technical paper presentations

SIGMOD (cont.)

18 industrial paper presentations
31 demonstrations
Best demonstration competition
New researchers symposium
Programming contest
Undergraduate posters competition
6 pre-conference workshops (DaMon, DBTest, IDAR, KEYS, MobiDE, WebDB)

PODS

Keynote presentation
- "Web Information Management: A Plethora of Problems"
  Raghu Ramakrishnan (Yahoo! Research)

2 Tutorials
- "The finite model theory toolbox of a database theoretician"
  Leonid Libkin (University of Edinburgh)
- "Worst-Case Efficient Range Search Indexing"
  Lars Arge (University of Aarhus)

26 technical paper presentations

We look forward to seeing you at SIGMOD/PODS 2009 in Providence.

ORGANIZATION

SIGMOD General Co-Chairs
Ugur Cetintemel & Stan Zdonik (Brown University)

PODS General Chair
Jan Paredaens (University of Antwerp)

SIGMOD PC Chair
Donald Kossman (28msec & ETH Zurich)

PODS PC Chair
Jianwen Su (UC Santa Barbara)

SIGMOD Publicity Chair
Yanlei Diao (UMass Amherst)

PODS Publicity and Proceedings Chair
Yi Chen (Arizona State University)

SIGMOD Proceedings Chair
Nesime Tatbul (ETH Zurich)

SIGMOD Industrial Papers Chairs
Michael Franklin (UC Berkeley & Truviso) & Donovan Schneider (Yahoo!)

SIGMOD Tutorials Chair
Sihem Amer-Yahia (Yahoo!)

SIGMOD Panel Chair
Jennifer Widom (Stanford)

SIGMOD Demo Chair
Björn Jónsson (Reykjavik University)

SIGMOD Exhibits Chair
Samuel Madden (Massachusetts Institute of Technology)

SIGMOD New Researcher Symposium Chairs
Christoph Koch (Cornell) & Nesime Tatbul (ETH Zurich)

SIGMOD Information Chair
Murali Mani (Worcester Polytechnic Institute)

Local Arrangements Chair
Daniel Abadi (Yale University)
Olga Papaemmanouil (Brandeis University)

Local Workshop Chair
Cindy Chen (UMass Lowell)

Web Chair
Gerome Miklau (UMass Amherst)

Registration Chair
Kajal Claypool (MIT Lincoln Labs)

Industrial sponsorship
Renee Miller (University of Toronto)

Finance Chair
Elke Rundensteiner (Worcester Polytechnic Institute)

SIGMOD Undergraduate Research Poster Competition Chairs
Laura Chiticariu (IBM Almaden Research Center)
Lukasz Golab (ATT Labs-Research)

SIGMOD Programming Contest Chairs
Samuel Madden (MIT) & Michael Stonebraker (MIT)

KEYS 2009

First International Workshop on Keyword Search on Structured Data

June 28, 2009, Providence, Rhode Island, USA

http://keys09.asu.edu

AIMS & TOPICS OF INTEREST

Information search is an indispensable component of our lives. Web search engines are widely used for searching textual documents, images, and video. However, there are also vast collections of structured and semi-structured data both on the Web and in enterprises, such as relational databases, XML data, etc. Traditionally, to access these resources, a user must learn structured or semi-structured query languages, and must be able to access data schemas, which are most likely heterogeneous, complex, and fast-evolving. To relieve web and scientific users from the learning curve and enable them to easily access structured and semi-structured data, there is a growing research interest to support keyword search on these data sources.

The aim of this workshop is to encourage researchers from both academia and industry communities to discuss the opportunities and challenges in keyword search on (semi-)structured data, and to present the key issues and novel techniques in this area. We invite researchers and practitioners working in relational databases, data warehouses, XML, information extraction, natural language processing, probabilistic databases, and related areas to attend this workshop. The main topics include but are not limited to:

- Keyword search on relational databases and data warehouses
- Keyword search on XML data
- Keyword search on extracted data from text documents
- Keyword on other data structures (e.g. workflows, annotated images)
- Keyword search on data streams and continuous monitoring systems
- Ranking schemes
- Top-K query processing
- Result snippet generation
- Result clustering
- Query cleaning
- User preferences and feedback
- Handling data uncertainty in keyword search
- Search quality evaluation
- Performance optimization

REGISTRATION

Registration for KEYS 2009 is done via the SIGMOD/PODS 2009 conference registration web site (https://regmaster2.com/cgi-bin/MOD09/on1/RMSs.cgi).

KEYNOTE TALKS

Structured Data and Web Documents: Better Together?
Surajit Chaudhuri (Microsoft Research)

Querying Text Databases and the Web: Beyond Traditional Keyword Search
Luis Gravano (Columbia University)

ACCEPTED PAPERS

- Research Papers

Do We Mean the Same? Disambiguation of Extracted Keyword Queries for Database Search
Elena Demidova, Irina Oelze, Peter Fankhauser (L3S Research Center)

Keyword Query Cleaning using Hidden Markov Models
Ken Q Pu (University of Ontario Inst. of Technology)

Efficient Top-k Algorithms for Fuzzy Search in String Collections
Rares Vernica, Chen Li (University of California, Irvine)

Hierarchical Result Views for Keyword Queries over Relational Databases
Shiyuan Wang (UC Santa Barbara), Junichi Tatemura, Arsany Sawires, Oliver Po (NEC Laboratories America), Divyakant Agrawal, Amr El Abbadi (UC Santa Barbara)

Query Segmentation Using Conditional Random Fields
Xiaohui Yu and Huxia Shi (York University)

A First Study on Strategies for Generating Workflow Snippets
Tommy Ellkvist, Lena Strömbäck (Linköping University), Lauro Didier Lins, Juliana Freire (University of Utah)

- Demos and Posters

Building Search Applications with MarkLogic Server (Invited Demo)
Ron Avnur (Mark Logic Corporation)

Language-Model-Based Ranking in Entity-Relation Graphs
Shady Elbassuoni, Maya Ramanath, Gerhard Weikum (Max-Planck Institute for Informatics)

Generic and Effective Semi-structured Keyword Search
Arash Termehchy, Marianne Winslett (University of Illinois, Urbana)

XML Keyword Query Refinement
Jiaheng Lu, Zhifeng Bao (National University of Singapore), Tok Wang Ling, Xiaofeng Meng (Renmin University of China)

DBDOC: Querying and Browsing Interrelated Documents in SQL
Carlos Garcia-Alvarado, Carlos Ordonez, Zhibo Chen (University of Houston)

CALL FOR PARTICIPATION

Registration Website
http://www.regmaster.com/conf/sigmod2009.html

Sponsored by

In-cooperation with

MobiDE 2009: Eighth International ACM Workshop on Data Engineering for Wireless and Mobile Access

10 Year Anniversary

June 29, 2009, Providence, RI, USA
(in conjunction with SIGMOD/PODS 2009)
http://www.cs.fsu.edu/mobide09/

Sponsors:

COOPERATING OBJECTS NETWORK OF EXCELLENCE

Microsoft .net Club

General Chairs:

Yannis Kotidis
Athens University of Economics and Business
kotidis@aueb.gr

Pedro Jose Marron
University of Bonn
pjmarron@cs.uni-bonn.de

Program Chairs:
Le Gruenwald
University of Oklahoma
ggruenwald@ou.edu

Demetris Zeinalipour
University of Cyprus
dzeina@cs.ucy.ac.cy

Publicity Chair:

Feifei Li
Florida State University
lifeifei@cs.fsu.edu

Demonstration Chair:

Zografoula Vagena
Microsoft Research Cambridge
zografv@microsfot.com

This is the eighth of a successful series of workshops that aims to act as a bridge between the data management, wireless networking, and mobile computing communities. This year's event marks the 10-year anniversary of the workshop. As in the past 10 years, the workshop will continue to serve as a forum for researchers and technologists to discuss the state-of-the-art, present their contributions, and set future directions in data management for mobile and wireless access.

The topics of interest related to mobile and wireless data engineering include, but are not limited to:

* ad-hoc networked databases
* consistency maintenance and management
* context-aware data access and query processing
* data caching, replication and view materialization
* data publication modes: push, broadcast, and multicast
* database issues for moving objects: storing, indexing, etc.
* mobile agent models and languages
* mobility-aware data mining and warehousing
* mobile database security
* mobile databases in scientific, medical and engineering applications
* mobile peer-to-peer applications and services
* mobile transaction models and management
* mobile web services
* mobility awareness and adaptability
* pervasive computing
* prototype design of mobile databases
* quality of service for mobile databases
* sensor network databases
* transaction migration, recovery and commit processing
* wireless multimedia systems

This year's program includes six research talks, two demos, two keynote speeches, and a panel discussion. The first keynote will be presented by Prof. Matt Welsh from Harvard University. The second keynote will be given by Dr. Frank Olken, a program director from National Science Foundation, Information and Intelligent Systems Division. The panel's title is "20 Years of Mobile Data Management Research: Vision and Reality", which will be moderated by Prof. Panos K. Chrysanthis from University of Pittsburgh. The program for the workshop is attached. We are looking forward to an exciting program and your participation!

Please register at http://www.regmaster.com/conf/sigmod2009.html

PRELIMINARY PROGRAM

Opening and Keynote One

Keynote Speaker: Matt Welsh (Harvard University, USA)
Keynote Title: "A New Era of Resource Responsibility for Sensor Networks"

Coffee Break (Foyer)

Session 1: Database Issues for Mobile Computing

"Using Transaction Isolation Levels for Ensuring Replicated Database Consistency in Mobile Computing Environments", Jose Monteiro (PUC-Rio, Brazil), Angelo Brayner (University of Fortaleza, Brazil) and Sergio Lifschitz (PUC-Rio, Brazil)

"Data-aware connectivity in mobile replicated systems", Joao Barreto, Joao Garcia, Luis Veiga, Paulo Ferreira (Technical University Lisbon, Portugal)

"Bandwidth-constrained Distributed Skyline Computation", Vlachou Akrivi and Kjetil Norvag (NTNU, Norway)

Lunch Break

Session 2: Context/Location-based Data Access and Query Processing

"Towards Context and Preference-Aware Location-based Database Systems", Mokbel Mohamed and Justin Levandoski (Univ. Of Minnesota, USA)

"Law-Aware Access Control for International Financial Environment", Stieghahn Michael and Thomas Engel (Univ. Of Luxembourg, Luxembourg)

"An Innovative Architecture for Context Foraging", Tsetsos Vassileios and Stathes Hadjiefthymiades (Univ. Of Athens, Greece)

Session 3: Demonstrations

"Demonstrating Evacuation Algorithms with Mobile Devices using an e-Scavenger Hunt Game", Connor Alexander, Callen Shaw, Alexander Connor, Alexandros Labrinidis and Panos Chrysanthis (Univ. of Pittsburgh, USA)

"MobiSNA: a Mobile Video Social Network Application", Liang Gou, Jung-Hyun Kim, Hung-Hsuan Chen (Penn State University, USA), Jason Collins, Marc Goodman (Alcatel-Lucent, USA), Xiaolong (Luke) Zhang, C. Lee Giles (Penn State University, USA))

Coffee Break (Foyer)

Keynote Two & Panel

Keynote Speaker: Dr. Frank Olken (Program Director, NSF, IIS Division),
Keynote Title: "Space, Time, Sensors, and Data Semantics"

Panel Title: "20 Years of Mobile Data Management Research: Vision and Reality"
Panel Moderator: Panos K. Chrysanthis (University of Pittsburgh)

Closing Remarks

12th International Workshop on the Web and Databases (WebDB 2009)

Providence, Rhode Island - June 28, 2009

Co-located with ACM SIGMOD 2009

Workshop Chairs

Alexandros Labrinidis
University of Pittsburgh, USA
Michalis Petropoulos
SUNY Buffalo, USA

Program Committee

Karl Aberer EPFL, Switzerland **Sihem Amer-Yahia** Yahoo!, US **Andrey Balmin** IBM Almaden, US **Denilson Barbosa** University of Alberta, Canada **Michael Benedikt** University of Oxford, UK **José A. Blakeley** Microsoft, US **Michael Carey** UC Irvine, US **Kevin Chen-Chuan Chang** UIUC, US; **Vassilis Christophides** ICS-FORTH, Greece **Alin Deutsch** UC San Diego, US **AnHai Doan** University of Wisconsin-Madison, US **Peter Dolog** Aalborg University, Denmark **Luna Dong** AT&T Labs, US **Schahram Dustdar** TU Vienna, Austria **Piero Fraternalli** Politecnico di Milano, Italy **Juliana Freire** University of Utah, US **Luis Gravano** Columbia, US **Vagelis Hristidis** Florida International University, US **Rick Hull** IBM T.J. Watson, US **Bertram Ludaescher** UC Davis, US **Qiong Luo** HKUST, Hong Kong **Jayant Madhavan** Google, US **Amelie Marian** Rutgers University, US **Paolo Merialdo** Universita di Roma Tre, Italy **Gerome Miklau** UMass-Amherst, US **Chris Olston** Yahoo!, US **Yannis Papakonstantinou** UC San Diego, app2you.com, US **Jayavel Shanmugasundaram** Yahoo!, US **Wang-Chiew Tan** UC Santa Cruz, US **Vasilis Vassalos** AUEB, Greece **Yannis Velegrakis** University of Trento, Italy **Jeffrey Yu** Chinese University of Hong Kong, Hong Kong

Web Chair

Demian Lessa
SUNY Buffalo, USA

Call For Participation

Keynote Talk

Title: **Do-It-Yourself Custom Database-Driven Web Applications**
Speaker: **Yannis Papakonstantinou** UC San Diego & app2you.com, US

Accepted Papers

» Extracting Route Directions from Web Pages
Xiao Zhang, Prasenjit Mitra, Sen Xu, Anuj Jaiswal, Alex Klippel, Alan MacEachren, Pennsylvania State University
» Querying DAG-shaped Execution Traces Through Views
Maya Ben-Ari, Tova Milo, Eldad Verbin, Tel Aviv University
» Efficient and Scalable Sequence-Based XML Filtering
Mariam Salloum, Vassilis Tsotras, UC Riverside
» Functional Dependency Generation and Applications in Pay-As-You-Go Data Integration Systems
Daisy Zhe Wang, UC Berkeley, Luna Dong, AT&T Labs, Anish Das Sarma, Stanford University, Michael Franklin, UC Berkeley, Alon Halevy, Google Inc.
» Bridging the Terminology Gap in Web Archive Search
Klaus Berberich, Srikanta Bedathur, Mauro Sozio, Gerhard Weikum, MPII
» Event Identification in Social Media
Hila Becker, Columbia University, Mor Naaman, Rutgers University, Luis Gravano, Columbia University
» A Machine Learning Approach to Foreign Key Discovery
Alexandra Rostin, Oliver Albrecht, Humboldt-Universität zu Berlin, Jana Bauckmann, Felix Naumann, Hasso-Plattner-Institut Potsdam, Ulf Leser, Humboldt-Universität zu Berlin
» Towards Well-Behaved Schema Evolution
Rada Chirkova, North Carolina State University, George Fletcher, Washington State University
» PrivatePond: Outsourced Management of Web Corpuses
Daniel Fabbri, Kristen LeFevre, Arnab Nandi, H. V. Jagadish, University of Michigan
» Experimental Evaluation of Query Processing Techniques over Multiversion XML Documents
Adam Woss, Vassilis Tsotras, UC Riverside
» Beyond the Stars: Improving Rating Predictions using Review Text Content
Gayatree Ganu, Rutgers University, Noémie Elhadad, Columbia University, Amelie Marian, Rutgers University

Accepted Demos

» Design Specification Techniques for Do-It-Yourself Application Platforms
Gaurav Bhatia, Yupeng Fu, UC San Diego, Keith Kowalczykowski, app2you.com, US, Kian Win Ong, Kevin Keliang Zhao, Yannis Papakonstantinou, Alin Deutsch, UC San Diego
» Entity Search with NECESSITY
Ekaterini Ioannou, L3S Research Center, Saket Sathe, Nicolas Bonvin, Anshul Jain, Srikanth Bondalapati, Gleb Skobeltsyn, Ecole Polytechnique Fédérale de Lausanne (EPFL), Claudia Niederée, L3S Research Center, Zoltan Miklos, Ecole Polytechnique Fédérale de Lausanne (EPFL)
» A Referential Integrity Browser for Distributed Databases
Carlos Ordonez, University of Houston, Javier Garcia-Garcia, Rene Villeda-Ruz, UNAM, Mexico

http://webdb09.cse.buffalo.edu

Gramado
Brazil

28th International Conference on Conceptual Modeling

CALL FOR PARTICIPATION

CONFERENCE ORGANIZATION

Conference General Chair
José Palazzo M. de Oliveira, UFRGS, BRA

Program Committee Co-Chairs
Alberto H. F. Laender, UFMG, BRA
Silvana Castano, UNIMI, ITA
Umesh Dayal, HP Labs, USA

Steering Committee Liaison
Arne Sølvberg, NTNU, NOR

Local Arrangements Chair
José Valdeni de Lima, UFRGS, BRA

Workshop Chairs
Carlos A. Heuser, UFRGS, BRA
Günther Pernul, U. Regensburg, GER

Tutorial Chairs
Daniel Schwabe, PUC-Rio, BRA
Stephen W. Liddle, BYU, USA

Panel Chair
David Embley, BYU, USA

Industrial Chair
Fabio Casati, U Trento, ITA

Demos and Posters Chairs
Altigran S. da Silva, UFAM, BRA
Juan-Carlos Mondéjar, U. Alicante, ESN

PhD Colloquium Chairs
Stefano Spaccapietra, EPFL, SWI
Giancarlo Guizzardi, UFES, BRA

Financial Chair
Renata de Matos Galante, UFRGS, BRA

Proceedings Chair
Daniela Musa, UNIFESP, BRA

Publicity Chair
Mirella M. Moro, UFMG, BRA

EARLY BIRD REGISTRATION
Until September 10, 2009.

WEBSITE
http://www.inf.ufrgs.br/ER2009

The International Conference on Conceptual Modeling is a leading international forum for presenting and discussing current research and applications in which the major emphasis is on conceptual modeling. Topics of interest span the entire spectrum of conceptual modeling including research and practice in areas such as theories of concepts and ontologies underlying conceptual modeling, methods and tools for developing and communicating conceptual models, and techniques for transforming conceptual models into effective implementations.

KEYNOTE SPEAKERS

Antonio L. Furtado, PUC-Rio, BRA
John Mylopoulos, UToronto, CAN
Laura Haas, IBM Almaden Research Center, USA

CONFERENCE VENUE

ER 2009 will be held at the beautiful city of **Gramado**, **Brazil**. Gramado is a small touristic town in the southern Brazilian state of Rio Grande do Sul. It is much more than a beautiful scenery, with natural landscapes of all colors and climates. It offers a perfect infra-structure to serve those who seek relaxation, pleasure, or adventure. This region offers visitors the best tourist infra-structure in the state. Cooking is rich and varied. Takes include irresistible chocolate, typical dishes, the plentiful colonial breakfast, and the taste of the best international cuisines.

WORKSHOPS

- ACM-L: Active Conceptual Modeling of Learning
- Conceptual Modeling in the Large
- ETheCoM: Evolving Theories of Conceptual Modelling
- FP-UML: Workshop on Foundations and Practices of UML
- LbM: Logic Based Modeling
- M2AS'09: Intl. Work. on Modeling Mobile Applications and Services
- MOST-ONISW: Joint Intl. Work. on Metamodels, Ontologies, Semantic Technologies, and Information Systems for the Semantic Web
- QoIS: Quality of Information Systems
- RIGiM: Requirements, Intentions and Goals in Conceptual Modeling
- SeCoGIS 2009: Semantic and Conceptual Issues in Geographic Information Systems