SIGMOD Officers, Committees, and Awardees

Chair	Vice-Chair	Secretary/Treasurer
Yannis Ioannidis	Christian S. Jensen	Alexandros Labrinidis
University of Athens	Department of Computer Science	Department of Computer Science
Department of Informatics	Aalborg University	University of Pittsburgh
Panepistimioupolis, Informatics Bldg	Selma Lagerlöfs Vej 300	Pittsburgh, PA 15260-9161
157 84 Ilissia, Athens	DK-9220 Aalborg Øst	
HELLAS	DENMARK	USA
+30 210 727 5224	+45 99 40 89 00	+1 412 624 8843
<yannis AT di.uoa.gr>	<csj AT cs.aau.dk >	<labrinid AT cs.pitt.edu>

SIGMOD Executive Committee:

Sihem Amer-Yahia, Curtis Dyreson, Christian S. Jensen, Yannis Ioannidis, Alexandros Labrinidis, Maurizio Lenzerini, Ioana Manolescu, Lisa Singh, Raghu Ramakrishnan, and Jeffrey Xu Yu.

Advisory Board:

Raghu Ramakrishnan (Chair), Yahoo! Research, <First8CharsOfLastName AT yahoo-inc.com>, Rakesh Agrawal, Phil Bernstein, Peter Buneman, David DeWitt, Hector Garcia-Molina, Masaru Kitsuregawa, Jiawei Han, Alberto Laender, Tamer Özsu, Krithi Ramamritham, Hans-Jörg Schek, Rick Snodgrass, and Gerhard Weikum.

Information Director:

Jeffrey Xu Yu, The Chinese University of Hong Kong, <yu AT se.cuhk.edu.hk>

Associate Information Directors:

Marcelo Arenas, Denilson Barbosa, Ugur Cetintemel, Manfred Jeusfeld, Dongwon Lee, Michael Ley, Rachel Pottinger, Altigran Soares da Silva, and Jun Yang.

SIGMOD Record Editor:

Ioana Manolescu, INRIA Saclay, <ioana.manolescu AT inria.fr>

SIGMOD Record Associate Editors:

Magdalena Balazinska, Denilson Barbosa, Chee Yong Chan, Ugur Çetintemel, Brian Cooper, Cesar Galindo-Legaria, Leonid Libkin, and Marianne Winslett.

SIGMOD DiSC and SIGMOD Anthology Editor:

Curtis Dyreson, Washington State University, <cdyreson AT eecs.wsu.edu>

SIGMOD Conference Coordinators:

Sihem Amer-Yahia, Yahoo! Research, <sihemameryahia AT acm.org> and Lisa Singh, Georgetown University, <singh AT cs.georgetown.edu>

PODS Executive: Maurizio Lenzerini (Chair), University of Roma 1, <lenzerini AT dis.uniroma1.it>, Georg Gottlob, Phokion G. Kolaitis, Leonid Libkin, Jan Paradaens, and Jianwen Su.

Sister Society Liaisons:

Raghu Ramakhrishnan (SIGKDD), Yannis Ioannidis (EDBT Endowment).

Awards Committee:

Laura Haas (Chair), IBM Almaden Research Center, <laura AT almaden.ibm.com>, Rakesh Agrawal, Peter Buneman, and Masaru Kitsuregawa.

Jim Gray Doctoral Dissertation Award Committee:

Johannes Gehrke (Co-chair), Cornell Univ.; Beng Chin Ooi (Co-chair), National Univ. of Singapore, Alfons Kemper, Hank Korth, Alberto Laender, Boon Thau Loo, Timos Sellis, and Kyu-Young Whang

[Last updated : Oct 12th, 2010]

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)
Masaru Kitsuregawa (2009)
Umeshwar Dayal (2010)

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)
Beng Chin Ooi (2009)
David Lomet (2010)

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:

- **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
- **2008** *Winner*: Ariel Fuxman, University of Toronto. *Honorable Mentions*: Cong Yu, University of Michigan; Nilesh Dalvi, University of Washington.
- **2009** *Winner*: Daniel Abadi, MIT. *Honorable Mentions*: Bee-Chung Chen, University of Wisconsin at Madison; Ashwin Machanavajjhala, Cornell University.
- **2010** *Winner:* Christopher Ré (advisor: Dan Suciu), University of Washington

Honorable Mentions: Soumyadeb Mitra (advisor: Marianne Winslett), University of Illinois, Urbana-Champaign; Fabian Suchanek (advisor: Gerhard Weikum), Max-Planck Institute for Informatics

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

[Last updated : Oct 12th, 2010]

Editor's Notes

Welcome to the June 2010 issue of the ACM SIGMOD Record.

The issue starts with the 2010 SIGMOD/PODS awards: the SIGMOD Edward F. Codd innovation award, the SIGMOD contribution award, the SIGMOD and PODS test-of-time Awards, the SIGMOD and PODS best paper awards, the SIGMOD Jim Gray dissertation awards, and the SIGMOD undegraduate posters award.

The issue features two regular research articles. The article by Bellahsene, Benbernou, Jaudouin, Pinet, Pivert and Toumani describes FORUM, a data integration project tailored at mediation in large and dynamic environments. The article by Cho, Lee, Hwang, Han and Lee focuses on the efficient implementation of the dominance relationship between tuples, by exploiting data-level parallelism in SIMD architectures. The resulting efficient operation can be used to enhance the performance of any skyline algorithm.

In the Database Principles column, Deutsch and Milo focus on probabilistic processes, commonly used to model business processes, e-commerce and Web application etc. They revisit the model for probabilistic processes established in their prior work, describe its expressivity and complexity, and provide a detailed comparison with other models proposed in this area.

The survey by Grabova, Darmont, Chauchat and Zolotaryova considers the needs for decision support tools within small and medium enterprises, for which the costs associating to the ownership and maintenance of a traditional OLAP architecture are prohibitive. The authors analyze and compare several classes of software (and in particular Web-based open source software) which could be used for decision support in this setting.

The Distinguished Profiles column is represented in this issue by an interview with Professor Elisa Bertino, speaking out on her co-authors and students, research work and funding in Italy, Europe more generally, and the United States, her favorite research results, the research that she would have liked to do, and more.

Our issue closes with two reports associated to the 2010 SIGMOD conference. Tao summarizes his experience as the Chair of the SIGMOD 2010 demonstration track, and reports on selecting and setting up the demo, and picking the best demonstrations at the conference. Genzmer, Hudlet, Park, Schall and Senellart report on organizing and running the SIGMOD 2010 programming contest, whose topic this year involved building an efficient distributed data management engine.

From our side on the SIGMOD Record editorial board (http://www.sigmod.org/publications/sigmod-record/editors) we have several news to announce to the community. First, we welcome two new associate editors: Chee Yong Chan, and Denilson Barbosa (also in charge of the Web edition).
Second, as promised in the last issue, we have devised an ***editorial policy for the SIGMOD Record***, outlined below.

General policy *SIGMOD Record publishes content that provides substantial value to the global database community and that is not appropriate for the community's conferences and regular journals.*
We list below the main categories of welcomed contributions, and our policy for processing them.

Announcements The Record is the newsletter of the ACM's SIGMOD chapter, and as such, one of its main missions is to disseminate interesting information to the community. Announcements of interest for the SIGMOD Record include calls for papers and calls for participation to workshops and conferences (especially, but not limited to, ACM or SIGMOD-related ones), book announcements, announcements of new initiatives, contests or competitions within or related to the international database research community etc.

Workshop reports Organizers of workshops on topics of interest to the database community in the large may want to alert the community to new trends and new research opportunities. We encourage workshop reports to be relatively concise (4 pages is an upper limit, whereas 2 pages is typically sufficient). Potential authors of workshop reports are strongly encouraged to perform a synthesis over the individual workshop contributions, possibly listing ideas not included, or not obvious, in any of the workshop contributions considered in isolation. A workshop reports too close to a table of contents is not more informative than the workshop's Web page, and therefore will not be accepted.

Research centers The goal of this column is to publicize academic and industrial research groups working on data and information management issues. An ideal article for this column depicts the anatomy of a group (or a few closely-tied groups), including a brief discussion of the group's history, its primary investigators, and recent activities along with collaborators and sources of funding. While most articles are invited by the research center editor, direct submissions are also solicited and encouraged.

Database Principles and Distinguished Profiles in Databases These columns are under the care of their respective editors, and contributions appear here only by invitation from the editor.

Surveys State-of-the-art surveys have a long successful tradition in the SIGMOD Record. We welcome short surveys (of up to 10 pages) on topics of interest for the database community. The size constraint should not, however, lead to incomplete or partial snapshots of the state of the art in a given area. Instead, we encourage a sharp and focused writing style, which may sacrifice detail or extensive examples for breadth and comprehensiveness.

Systems and prototypes The SIGMOD Record welcomes short papers (up to 6 pages) describing innovative systems and prototypes. Such papers may (but do not need to) be developed versions of papers describing system demonstrations in a conference. However, demo papers are not acceptable as such. We encourage authors to omit details on GUIs (unless user interaction is the main topic of the system), demonstration scenario and such, and instead provide more insight into how the system is built, what are its internal modules, and why it has been done this way.

Regular articles The Record welcomes short research articles (of up to 6 pages), subject to the usual novelty and technical soundness requirements of mainstream conferences and workshops. By its nature, the Record cannot and will not, however, serve as an alternative for regular publication venues such as conferences, journals, or workshops. Regular articles welcome to the Record should fit broadly in one of these three categories:

- **Vision papers**, outlining new ideas or lines of research, not necessarily well-explored at the time of the writing, if they are deemed original, compelling, and of interest to the community at large
- **Regular papers**, structured and organized the way conference and workshop papers are, but whose natural length (when all the necessary ingredients of the work have been well presented) is of at most 6 pages. This page limit is (much) shorter than for regular conference papers, and set on purpose to materialize the different type of contributions that we expect in the Record.
- **Project reports**, describing a (typically large) R&D project carried on within one or several institutions, over a period of one to a few years. Such reports are welcome provided that they

describe innovative work, and are strictly more informative than "the sum of all the projects' parts".

Other contributions As the (database) world evolves, we expect to host other categories of content, following suggestions and interesting ideas from the community. Please contact Ioana Manolescu (Ioana.Manolescu@inria.fr) if you have a proposal that does not fit any of the above.

Duplicate submissions Workshop reports, surveys, system and prototype papers, and regular papers should not be published, or under consideration in any other formal publication venue during the time period when they are under consideration for the SIGMOD record. A significant overlap (more than 25% of the SIGMOD Record submission and 1 common author) with such a previous or concurrent submission is not allowed either. Violation of the duplicate submission policy will typically result in outright rejecting the SIGMOD Record submission. Prospective authors are encouraged to signal any previous or concurrent publication closely related to the SIGMOD Record submission, in the Cover Letter form of the Web based submission tool (http://db.cs.pitt.edu/recess/), and explain there why the submission does not violate the duplicate submission policy.

How to submit Submissions should be made electronically via the RECESS tool (http://db.cs.pitt.edu/recess/). Authors must ensure that their submission is properly formatted according to the guidelines (http://www.sigmod.org/publications/sigmod-record/authors).

Processing submissions Submissions to the SIGMOD Record are processed as follows.

- *Announcements* are processed by one editor, which accepts or rejects them, and may request modifications before accepting them.
- *Workshop reports* are processed by the reports editor, who may ask for modifications to be made in revised versions.
- *Surveys*, and *system and prototype papers*, are processed by the respective editor, who may solicit the opinions of external reviewers. The survey editor may then recommend acceptance with or without revision, or rejection.
- *Regular articles* are assigned by the editor-in-chief to an associate editor, who may solicit the opinion of external reviewers, recommend acceptance with or without revision, or rejection.

Ioana Manolescu

October 2010

Past SIGMOD Record Editors:

Harrison R. Morse (1969)
Daniel O'Connell (1971 – 1973)
Randall Rustin (1975)
Thomas J. Cook (1981 – 1983)
Jon D. Clark (1984 – 1985)
Margaret H. Dunham (1986 – 1988)
Arie Segev (1989 – 1995)
Jennifer Widom (1995 – 1996)
Michael Franklin (1996 – 2000)
Ling Liu (2000 – 2004)
Mario Nascimento (2005 – 2007)
Alexandros Labrinidis (2007 – 2009)

SIGMOD/PODS 2010 Awards

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.

Dr. Umeshwar Dayal is the recipient of the 2010 SIGMOD *Edgar F. Codd Innovation Award* for a succession of pioneering, influential contributions in distributed heterogeneous databases, high-performance active databases, generalized transitive closure, transaction models for long-running activities, and business process discovery, among other topics.

Umeshwar Dayal has a track record of 30 years of research accomplishments in data management and has made a succession of fundamental contributions to the field. His research in the mid-1980's on Multibase, the world's first large-scale heterogeneous distributed database system, paved the way for research over the past two decades at universities and industrial research labs (HP Labs, IBM Research, and elsewhere) in data integration and federated databases. PROBE, one of the first extensible database management systems, made significant contributions to the field in several areas, including object and semantic data modeling, spatial, temporal and recursive query processing, and system architecture. HiPAC was one of the leading active database systems of its time, and the only one to focus on the needs of real-time applications, resulting in an innovative transaction model. The event-condition-action (ECA) rule model introduced in HiPAC has now been widely adopted in reactive computing systems, complex-event-processing systems, and distributed middleware.

Dayal also has significant results in query-processing research, with particularly strong contributions to the processing of multi-database queries, spatial queries, and recursive queries. The approach described in his VLDB 1987 paper on unnesting SQL queries was later adopted in at least five commercial products. In addition, Dayal performed pioneering work in long-duration transactions, business-process management, and database design. In particular, he pioneered the field of business-process intelligence, which combines data warehousing, data mining, analytics and optimization techniques to monitor, control, analyze, and optimize business processes. Over 160 research papers and over 25 patents testify to Dayal' s innovation and productivity.

In 2001, he received the prestigious 10-year best paper award from VLDB for his paper on a transactional model for long-running activities. He is an HP Fellow, recognized for career contributions that " caused substantial change" in the state of the art while also improving HP products. In addition to his many innovative technical contributions, he has a distinguished record of service to our community, as a member of the editorial board of major journals (including ACM TODS and VLDB), chairing conferences (including SIGMOD, VLDB and ICDE), serving on boards and steering committees (VLDB, IEEE TC on Electronic Commerce, SIAM International Conference on Data Mining among them), and mentoring junior colleagues and young researchers.

SIGMOD Contributions Award for outstanding and sustained services to and promotion of the database field through activities such as education, conference organization, journals, standards, and research funding.

Dr. David Lomet is the recipient of the 2010 SIGMOD Contribution Award for his outstanding leadership as the Editor-I n-Chief of the IEEE Data Engineering Bulletin, a key forum for dissemination of emerging ideas in academia and industry. Lomet has been a key figure in our field, holding many additional leadership roles and demonstrating in each his dedication toservice and to our community.

By awarding David Lomet the ACM Contributions Award, we recognize his outstanding contributions to our community in leading the I EEE Data Engineering Bulletin for nearly 20 years and thereby creating a collection of timely articles of great value. He has almost single-handedly driven the IEEE Data Engineering Bulletin, providing our community with a constant stream of special issues, assembled by world-class invited guest editors. This service has been a wonderful benefit to the field, as each issue has provided a "root node" into key projects, both academic and industrial, and into the research literature related to the topic of the issue. These articles and issues have thus provided a "fast path" to see what's happening in an area as well as a way to make sure industrial highlights are brought to the attention of academics and vice versa. Not only has Lomet run the Bulletin, but he also negotiated with IEEE Computer Society and authors to make issues of the Bulletin available on CDROM via the SIGMOD DiSC, and later, to digitize the entire set of issues from 1977 on so that they can be available online to all.

Lomet has made significant contributions to our field through service in other roles as well. He was a Co-PC Chair of VLDB and he is currently on the VLDB Board of Trustees. He has been both a Co-PC Chair and a Co-General Chair of IEEE Data Engineering Conferences, and served as a member of the Steering Committee of the IEEE Technical Committee on Data Engineering from 2004-2009. He sets a high standard of service for our community.

SIGMOD Test-of-Time Award for the paper from the 2000 SIGMOD Conference that has had the most impact (research, products, methodology) over the intervening decade.

NiagaraCQ: A Scalable Continuous Query System for Internet Databases (http://portal.acm.org/citation.cfm?doid=342009.335432)

Jianjun Chen, David J. DeWitt (University of Wisconsin, now Microsoft), Feng Tian (University of Wisconsin, now VMWare), Yuan Wang (University of Wisconsin, now Microsoft)

This paper from the SIGMOD 2000 Conference bridged from the world of continuous, or standing, queries against a changing stored database, to stream processing systems. NiagaraCQ was a pioneering system, the first to address the problem of the millions of overlapping queries that would need to be supported in a truly internet-scale system. It used relational-style operators to optimize a given set of continuous queries. Similar frameworks appeared in subsequent studies of stream databases, sensor databases, information delivery systems, and complex-event-processing (CEP) systems. The idea of dynamic optimization of continuous queries leveraging database operators (including dynamic query grouping and split) became a baseline for modern streaming data platforms. In summary, this paper helped open the new field of high-performance systems for continuous query processing, and was a strong force in shaping the following generations of stream processing systems.

Abstract: Continuous queries are persistent queries that allow users to receive new results when they become available. While continuous query systems can transform a passive web into an active environment, they need to be able to support millions of queries due to the scale of the I nternet. No existing systems have achieved this level of scalability. NiagaraCQ addresses this problem by grouping continuous queries based on the observation that many

web queries share similar structures. Grouped queries can share the common computation, tend to fit in memory and can reduce the I /O cost significantly. Furthermore, grouping on selection predicates can eliminate a large number of unnecessary query invocations. Our grouping technique is distinguished from previous group optimization approaches in the following ways. First, we use an incremental group optimization strategy with dynamic regrouping. New queries are added to existing query groups, without having to regroup already installed queries. Second, we use a query-split scheme that requires minimal changes to a general-purpose query engine. Third, NiagaraCQ groups both change-based and timer-based queries in a uniform way. To insure that NiagaraCQ is scalable, we have also employed other techniques including incremental evaluation of continuous queries, use of both pull and push models for detecting heterogeneous data source changes, and memory caching. This paper presents the design of NiagaraCQ system and gives some experimental results on the system's performance and scalability.

SIGMOD Best Paper Award

FAST: Fast Architecture Sensitive Tree Search on Modern CPUs and GPUs (http://portal.acm.org/citation.cfm?doid=1807167.1807206)

Changkyu Kim, Jatin Chhugani, Nadathur Satish (Intel), Eric Sedlar (Oracle), Anthony Nguyen (Intel), Tim Kaldewey (Oracle), Victor Lee (Intel), Scott Brandt (University of California, Santa Cruz), Pradeep Dubey (Intel)

This paper presents FAST, a layout for an in-memory binary tree index that is well-suited for state-of-the-art CPU and GPU architectures. The layout and associated search methods take advantage of SIMD instructions and thread-level parallelism (TLP). FAST also accounts for cache-line sizes and hides cache-miss and TLB-miss latency by processing many outstanding queries simultaneously (with software pipelining and TLP). The paper shows that with all these optimizations, search on GPU is compute bound and search on a CPU is bandwidth bound. To optimize the latter further, the paper presents a key-compression scheme, which also takes advantage of SIMD instructions, to alleviate bandwidth limits and handle larger keys. Experiments show how CPU and GPU perform on trees with different sizes, how many concurrent queries are needed to achieve their peak throughput, and how compression can improve search performance. This paper is an excellent research contribution that provides an end-to-end system design and associated algorithms and techniques to develop a complete solution that leverages the underlying hardware architecture. Given the modular structure of the overall design, the solution can easily be adapted to future architectures.

PODS Best Paper Award

An Optimal Algorithm for the Distinct Elements Problem (http://portal.acm.org/citation.cfm?doid=1807085.1807094)

Daniel M. Kane, Jelani Nelson and David P. Woodruff

Abstract: We give the first optimal algorithm for estimating the number of distinct elements in a data stream, closing a long line of theoretical research on this problem begun by Flajolet and Martin in their seminal paper in FOCS 1983. This problem has applications to query optimization, Internet routing, network topology, and data mining. For a stream of indices in $\{1,\ldots,n\}$, our algorithm computes a $(1 \pm \varepsilon)$-approximation using an optimal $O(\varepsilon^{-2} + \log(n))$ bits of space with 2/3 success probability, where $0<\varepsilon<1$ is given. This probability can be amplified by independent repetition. Furthermore, our algorithm processes each stream update in $O(1)$ worst-case time, and can report an estimate at any point midstream in $O(1)$ worst-case time, thus settling both the space and time complexities

simultaneously. We also give an algorithm to estimate the Hamming norm of a stream, a generalization of the number of distinct elements, which is useful in data cleaning, packet tracing, and database auditing. Our algorithm uses nearly optimal space, and has optimal *O(1)* update and reporting times.

The ACM PODS Alberto O. Mendelzon Test-of-Time Award 2010

In 2007, the PODS Executive Committee decided to establish a Test-of-Time Award, named after the late Alberto O. Mendelzon, in recognition of his scientific legacy, and his service and dedication to the database community. Mendelzon was an international leader in database theory, whose pioneering and fundamental work has inspired and influenced both database theoreticians and practitioners, and continues to be applied in a variety of advanced settings. He served the database community in many ways; in particular, he served as the General Chair of the PODS conference, and was instrumental in bringing together the PODS and SIGMOD conferences. He also was an outstanding educator, who guided the research of numerous doctoral students and postdoctoral fellows.

The Award is to be awarded each year to a paper or a small number of papers published in the PODS proceedings ten years prior, that had the most impact (in terms of research, methodology, or transfer to practice) over the intervening decade. The decision was approved by SIGMOD and the ACM. The funds for the Award were contributed by IBM Toronto.

After careful consideration, the Award Committee for 2010 has decided to select the following papers as the award winners for 2010:

- **Typechecking for XML Transformers,** Tova Milo, Dan Suciu, and Victor Vianu (http://doi.acm.org/10.1145/335168.335171)

  Abstract: We study the typechecking problem for XML transformers: given an XML transformation program and a DTD for the input XML documents, check whether every result of the program conforms to a specified output DTD. We model XML transformers using a novel device called a k-pebble transducer, that can express most queries without data-value joins in XML-QL, XSLT, and other XML query languages. Types are modeled by regular tree languages, a nobust extension of DTDs. The main result of the paper is that typechecking for k-pebble transducers is decidable. Consequently, typechecking can be performed for a broad range of XML transformation languages, including XML-QL and a fragment of XSLT.

- **Integrity Constraints for XML,** Wenfei Fan and Jérôme Siméon (http://doi.acm.org/10.1145/335168.335172)

  Abstract; Integrity constraints are useful for semantic specification, query optimization and data integration. The ID/IDREF mechanism provided by XML DTDs relics on a simple form of constraint to describe references. Yet, this mechanism is not sufficient to express semantic constraints, such as keys or inverse relationships, or stronger, object-style references. In this paper, we investigate integrity constraints for XML, both for semantic purposes and to improve its current reference mechanism. We extend DTDs with several families of constraints, including key, foreign key, inverse constraints and constraints specifying the semantics of object identities. These constraints are useful both for native XML documents and to preserve the semantics of data originating in relational or object databases. Complexity and axiomatization results are established for the (finite) implication problems associated with these constraints. These results also extend relational dependency theory on the interaction between (primary) keys and foreign keys. In addition, we investigate implication of more general constraints, such as functional, inclusion and inverse constraints defined in terms of navigation paths.

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.

The 2010 recipient is **Chris Re** (University of Washington, advised by Dan Suciu) for his thesis titled *"Managing Large-scale Probabilistic Databases"*.

The two runners-up are Soumyadeb Mitra, (University of Indiana at Urrbana Champaign, advised by Marianne Winslett), for his thesis titled *"Trustworthy and Cost Effective Management of Compliance Records",* and Fabian Suchanek (Max-Planck Institut fur Informatik, advised by Gerhard Weikum) for his thesis titled *"Automated Construction and Growth of a Large Ontology"*

SIGMOD undergraduate posters

SIGMOD conference has started a new initiative that provides undergraduate students an opportunity to showcase their research accomplishments in a poster competition. The recipients for 2010 are:

1. **Zhenqiang Gong,** University of Science and Technology of China (USTC) *Parallel Algorithms for Top-k Query Processing*
2. **Yael Amsterdamer,** Tel Aviv University *Top-k Algorithms for Interactive Processes*
3. **Ian Charlesworth,** University of Waterloo *Analyzing Plan Spaces of Query Optimizers*
4. **Bharath Vissapragada,** International Institute of Information Technology, Hyderabad (IIIT-H) *Query Optimization on Cloud*
5. **Thodoris Georgiou,** National and Kapodistrian University of Athens *Extracting Topics of Debate between Users on Web Discussion Boards*
6. **Manos Karvounis,** National and Kapodistrian University of Athens *Utilizing the Quoting System of Online Web Forums to Estimate User Agreement*

SIGMOD best undegraduate posters: Thodoris Georgiou and Manos Karvounis, National and Kapodistrian University of Athens

The **2010 SIGMOD programming contest**, as well as and the **best SIGMOD 2010 demonstration award**, are detailed in the respective events report of this SIGMOD Record issue.

FORUM: A Flexible Data Integration System Based on Data Semantics*

Zohra Bellahsene
LIRMM
Montpellier – France
bella@lirmm.fr

Salima Benbernou †
LIPADE
Paris – France
salima.benbernou@parisdescartes.fr

Hélène Jaudoin
IRISA-ENSSAT
Lannion – France
jaudoin@enssat.fr

Francois Pinet
CEMAGREF
Clermont-Ferrand – France
francois.pinet@cemagref.fr

Olivier Pivert
IRISA-ENSSAT
Lannion – France
pivert@enssat.fr

Farouk Toumani
LIMOS
Clermont-Ferrand – France
ftoumani@isima.fr

ABSTRACT

The FORUM project aims at extending existing data integration techniques in order to facilitate the development of mediation systems in large and dynamic environments. It is well known from the literature that a crucial point that hampers the development and wide adoption of mediation systems lies in the high entry and maintenance costs of such systems. To overcome these barriers, the FORUM project investigates three main research issues: (i) automatic discovery of semantic correspondences (ii) consistency maintenance of mappings, and (iii) tolerant rewriting of queries in the presence of approximate mappings.

1. INTRODUCTION

Integrating and sharing information across disparate data sources entails several challenges: relevant data objects are split across multiple sources, often owned by different organizations. The data sources represent, maintain, and export their data using a variety of formats, interfaces and semantics. In the last two decades, much research work in the database community has been devoted to developing approaches and systems that enable information integration in heterogeneous and distributed environments. The FORUM project aims at leveraging integration technology in order to facilitate flexible information integration over a large number of sources. More precisely, we focused on three issues:

- flexibility of integration, mainly by investigating approaches that enable automatic schema mapping discovery,
- flexibility at a semantic level, by investigating the mapping maintenance and restructuring according to the evolution of the data sources, and
- flexibility and scalability of query processing: our aim is the design and development of flexible query rewriting algorithms that scale up in the number of available information sources.

In order to alleviate the integration task, we developed techniques that enable automatic discovery of semantic mappings between schemas of heterogeneous information sources. We implemented a tool that combines existing matching discovery algorithms. The system exploits machine learning techniques in order to combine the most appropriate algorithms with application domains.

The maintenance of schema mappings is an issue that deserves specific attention, more particularly in the context of dynamic environments where source contents and schemas may evolve very frequently. In the FORUM project, we addressed the problem of maintaining consistency of mappings between XML data when changes in source schemas occur. We proposed a two-step approach that exploits regular grammar's tree to first identify schema changes that may affect existing mappings and then suggest adequate incremental adaptation of the affected mappings.

In open environments, it is unrealistic to assume that the mappings may always be precisely defined. Thus, we studied the problem of answering queries in the context of a mediation system equipped with approximate mappings obtained from a relaxation

*Supported by ANR Research Grant ANR-05-MMSA-0007

†The work has been conducted while she was at LIRIS-université de Lyon

of the constraints present in user queries. We developed a flexible query rewriting technique that exploits approximate mappings in order to compute tolerant rewritings. From a formal point of view, we moved from the classical query semantics based on the notion of *certain answers* to a new semantics based on a notion of *probable answers*. Moreover, since the number of approximate rewritings may be very high, we devised a rewriting algorithm that generates only the top-k rewritings of a given query.

A prototype implementing the approach has been developed and experimented using hundreds of real-world data sources from the agricultural domain. A mediator schema was produced as well as correspondences between the sources and the mediator schema in order to facilitate data exchange as well as the querying process.

The remainder of the paper is structured as follows. Section 2 presents an automatic approach to the discovery of dependences, based on the use of a decision tree. Section 3 deals with schema mapping maintenance and describes an incremental approach that decomposes complex changes into atomic ones. Section 4 describes the flexible rewriting technique which relies on an approximate matching of interval constraints. Section 5 presents an application of these works to the traceability of agricultural activities. Section 6 recalls the main contributions and concludes the paper.

2. AUTOMATIC DISCOVERY OF SEMANTIC CORRESPONDENCES

Schema matching is the task of discovering correspondences between semantically similar elements of two schemas or ontologies [5, 11, 12, 13]. While mappings between a source schema and target schema specifies how the data in each of the source schemas is to be transformed to targed schema. In this subsection, our emphasis is on schema matching as it has been discussed in the survey by Rahm and Bernstein [16], and extended by Shvaiko and Euzenat in [21] with respect to semantic aspects. We have designed several algorithms for schema matching particularly in a large scale context [18]. Due to space limitation, we only describe the most recent of these approaches.

The kernel of traditional matching tools is the aggregation function, which combines the similarity values computed by different similarity measures. However, the aggregation function entails several major drawbacks: running all similarity measures between all elements from input schemas is expensive in terms of time performance and it can negatively influence the quality of matches. Furthermore, adding new similarity measures mainly implies to update the aggregation function. Finally, a threshold is applied on the aggregated value; yet, each similarity measure has its own value distribution, thus each should have its own threshold. The novel approach we propose has been implemented as a protoype named MatchPlanner [6], avoids the aforementioned drawbacks. Its principle consists in replacing the aggregation function by a decision tree. Let us recall that decision trees are predictive models in which leaves represent classes and branches stand for conjunction of features leading to one class. In our context, a decision tree is a tree whose internal nodes represent the similarity measures, and the edges stand for conditions on the result of the similarity measure. Thus, the decision tree contains plans (i.e., ordered sequences) of similarity measures. All leaf nodes in the tree are either *true* or *false*, indicating if there is a correspondence or not. We use well-known similarity measures from Second String [20], i.e., Levenshtein, trigrams, Jaro-Winkler, etc. We also added the neighbor context from [7], an annotation-based similarity measure, a restriction similarity measure and some dictionary-based techniques [23].

A first consequence of using a decision tree is that the *performance* is improved since the complexity is bounded by the height of the tree. Thus, only a subset of the similarity measures it involves is used for an actual matching task. The second advantage lies in the improvement of the *quality* of matches. Indeed, for a given domain, only the most suitable similarity measures are used. Moreover, the decision tree is flexible since new similarity measures can be added, whatever their output (discrete or continuous values).

Let us now outline the different steps of the algorithm. The similarity value computed by a similarity measure must satisfy the condition (continuous or discrete) on the edges to access a next node. Thus, when matching two schema elements with the decision tree, the first similarity measure — that at the root node — is used and returns a similarity value. According to this value, the edge for which its condition is satisfied leads to the next tree node. This process iterates until a leaf node is reached, indicating whether the two elements match or not. The final similarity value between two elements is the last one which has been computed, since we consider that the previous similarity values have only been computed to find the most appropriate similarity measure. Figure (1) illustrates an example of a decision tree. Now, let us illustrate how the

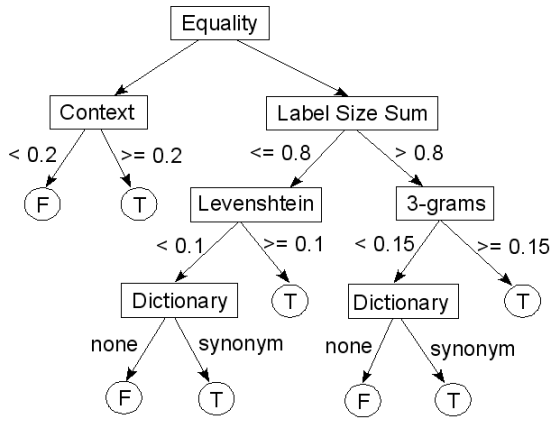


Figure 1: Example of decision tree

matching works using this tree and three pairs of labels to be compared.

- ***(quantity, amount)*** is first matched by *equality* which returns 0, then the *label sum size* is computed (value of 14), followed by the *3-grams* similarity measure. The similarity value obtained with 3-grams is low (0.1), implying the *dictionary* technique to be finally used to discover a synonym relationship.

- on the contrary, ***(type_of_culture, cultureType)*** is matched using *equality*, then *label sum size*, and finally *3-grams* which provides a sufficient similarity value (0.17) to stop the process.

- finally, the pair ***(analysis_date, analysis_date)*** involves identical labels, implying the *equality* measure to return 1. The *neighbor context* must then be computed to determine if there is a match or not. Indeed, both labels may refer to different analysis, for example one could represent a water analysis and the other stand for a report date.

Thus, only nine similarity measures have been used to discover the matches: four for *(quantity, amount)*, three for *(type_of_culture, cultureType)* and two for the last pair, instead of eighteen if all distinct similarity measures from the tree had been used. Indeed, with a matching tool based on an aggregation function, all of the similarity measures would have been applied for each pair of schema elements.

This approach has been implemented as a matching tool. Several default decision trees are provided and the user can select his/her configuration so as to emphasize e.g. the performance aspect or the quality of matches.

3. SCHEMA MAPPING MAINTENANCE

In dynamic networks such as peer to peer architectures, the nodes may change not only their data but also their schemas and their query domain. Facing this situation, schema mappings can become obsolete, a phase of maintenance is thus needed to maintain their consistency. Several solutions have been proposed to automate the adaptation of mappings when data schemes evolve. These solutions can be classified into two categories: (1) *incremental method* is implementing changes separately for each type of change occurs in the source and target schema [22]; (2) *Composition approach*, is a mapping-based representation of schema, and it is more flexible and expressive than the change-based representation [3].

Based on those categories, the proposed approach is not limited to the adaptation of affected matchings, but it also enables automatic detection of schema changes and decomposition of modifications into basic changes. It is two-phase approach:

1. processing on the schema to detect and automatically identify the changes,

2. processing on the mappings to provide an adaptation and keep the mappings consistent and compliant with the peer schemas.

Phase 1. We focused on handling the matchings for XML schemas. Then, the query languages for XML data model such as XQuery or XSLT are restricted to providing mechanisms for querying data sources, and the results are files in XML format. In this case, no device can be used to detect real time changes in the data schema. Facing situations that are most likely to cause changes, a first step is to determine if these treatments affect the structure and/or the semantics of the schema. Detecting a change in the schema level may then be interpreted as comparing two schemas (the old one and the new one), i.e., comparing two simple regular grammars' trees. This is possible since an *XML Schema* document can be represented by a regular grammar tree of a simple type (GARS) [14]. However, the existing GARS algorithms are similarity-metric-based approaches. In our case, the comparison between two GARS will provide elementary changes, and then should verify (1) syntactic equivalence (2) structural equivalence i.e., the tree generated using the first grammar must exhibit the same structure as the one generated by the second grammar. Due to space limitation, we just give an overview of the description.

The proposed algorithm will process in two steps, first it compares the terminal sets of both gram-

mars G_1 ,G_2 (associated with sources S_1 and S_2 resp), and the result is two sets t^1 and t^2 (terminals belonging to one grammar and not to the other), which is not enough to detect the changes. The second step will give more information by comparing each production rule with its juxtaposed rule, tackling the suppression, the renaming and the moving of an element. For instance, if an entity is moved, it will appear neither in t^1 nor in t^2, such change can only be detected by the production rules. The algorithm complexity is polynomial.

Phase 2. Once the detection is achieved, the next step is to report the modifications on the mapping in order to keep its semantics compliant. There exists different mapping representations based on a query language such as s-t-tgds[17]or [10], or very recent work in MapMerge prototype[2]. We used a mapping representation in a general way — a fragment of first order like language — that can be translated to any other existing representation. It including variables, constraints on the variables and the correspondences. Let us illustrate the mapping over our case study in the project — the agriculture area —, and consider the source schema S1 (*parcel 1* including an entity Info) and the target schema S2 (*parcel 2* including an entity Personal), one of the mappings is:

$$m ::= (\forall x \in S1.Info) \wedge (\exists y \in S2.Personal)) \wedge (y.Title = Director) \rightarrow [(x.NumFarmhand = y.SocialNum) \wedge (x.NameFarmHand = y.Name)]$$

It is represented by four parts and is interpreted as follows: (1) for each element x in the entity Info from the source $S1$ belonging to the mapping m, (2) it will exist elements y in the entity Personal (3) with a constraint on y (the title of the personal is Director — used to represent a restriction on the entities of the schemas) (4) there exist correspondences between variables defined by "=" function; *NumFarmhand* from the source $S1$ is equal to *SocialNum* from $S2$ and *NameFarmHand* ($S1$) equal to *Name* ($S2$).

An incremental approach is used in order to decompose complex changes into atomic ones. We classify them into *structural* (add, delete, modify an element of the tree), and *semantic* changes (add or delete, a correspondence constraint between two elements in the same schema or a key). Adding an element will not affect the mapping m, because it is a new information and we need to discover the matching between sources by means the algorithm presented in previous section. When deleting a complex element e (with many attributes), the adaptation in the mapping is limited to deleting the mapping assigned to e. If the deleted entity is atomic (one attribute) or belonging to a complex structure assigned to a mapping, the adaptation of m will be done over the substitution of clauses that refer to e in m (variable, conditions and correspondences). While displacing an entity (subtree), two adjustments are required:

- adaptation of the schema's internal constraints, thus, a redefinition of referential constraints must automatically be issued. Let e be the moving entity, o the origin of the reference and d its destination. There are three possible cases of adaptation of the reference during the displacement. In the first case, o and d belong to the same displaced structure; the reference remains unchanged. The second case describes a reference source $o \in e$ and its target $d \notin e$, the redefinition is achieved by taking into account the new location of o. Finally, if o does not belong to e, a new reference is set between o and d in its new location.
- adaptation of the mappings related to the displaced entity e. If e is complex, the adaptation is limited only to update the location, i.e., the variable access. If e is atomic, one needs to split the mapping and assign the element to a new mapping with constraints and correspondences. Renaming an element implies renaming all occurrences of e in the mapping. The referential changes (correspondences) will affect only the destination schema.

A prototype of the adaptation part has been developed towards the use case in agriculture.

4. FLEXIBLE QUERY REWRITING

In this section, we deal with flexible query answering in the FORUM data integration system. A Local-As-View type of approach is used, i.e., data sources are defined as views over the global schema. We consider the case where views and queries involve simple interval constraints $X \in [a, b]$ where X is an attribute name and a, b are two constants. The problem of rewriting queries using views in the presence of interval constraints is well known [15, 1]. However, in this section, we investigate this problem by means of a *tolerant* method. As an example, let us consider a query Q that aims to retrieve *codes* of *OMSParcels* (*organic matter spreading parcels*, i.e., agricultural plots receiving organic matter) whose *surface* is in [22, 35] hectares, and two views V_1 and V_2 such that:

V_1 provides *codes* of *OMSParcels* whose *surface* $\in [20, 40]$
V_2 provides *codes* of *OMSParcels* whose *surface* $\in [30, 45]$.

Both V_1 and V_2 have an interval constraint on attribute *surface*. However, none of these intervals is included in that of the query, thus the mappings between the two intervals and that of the query are only partial (imperfect). Moreover, since V_1 and V_2 only provide codes of parcels, selection on attribute surface is impossible, hence V_1 and V_2 cannot be used to get certain answers to Q. In such a context, rewriting algorithms based on the *certain answer* semantics fail to reformulate the query, thus to provide the user with any answer. The idea we advocate is to exploit approximate mappings between interval constraints involved in views and queries in order to compute *tolerant* query rewritings. Any such rewriting Q' is associated with a score between 0 and 1 which reflects the probability for a tuple returned by Q' to satisfy the initial query Q.

One considers any candidate rewriting, i.e., a rewriting which is contained in Q when interval constraints are ignored. Then one computes an *inclusion degree* between each pair of intervals I'_Q and I_Q restricting a same attribute X as follows:

$$deg(I_{Q'} \subseteq_{tol} I_Q) = \frac{\mid I_{Q'} \cap I_Q \mid}{\mid I_{Q'} \mid} \qquad (1)$$

This degree corresponds to the proportion of elements from $I_{Q'}$ which are in $I_{Q'} \cap I_Q$ when the distribution of the values over the domain is continuous and uniform.

Let us come back to the previous example and consider the intervals $I_{V_1} = [20, 40]$ and $I_Q = [22, 35]$ respectively involved in view V_1 and query Q. The inclusion degree between I_{V_1} and I_Q equals $\alpha = \frac{|[22,35]|}{|[20,40]|} = \frac{13}{20} = 0.65$. We can then attach this degree to the rewriting V_1 of Q, which is denoted by $V_1 \sqsubseteq_{0.65} Q$. According to the semantics of Equation 1, it means that an answer to V_1 has the probability 0.65 to satisfy the constraints of Q.

When the query involves several constraints, the degrees obtained are aggregated in order to calculate the global score associated with Q'. Notice that the inclusion degree associated with an attribute X denotes the probability that an answer to Q' satisfy the constraint over X in Q. Then the global score expresses the *probability* for an answer returned by the rewriting Q' to be an answer to Q. When constraints are attached with *existential* variables in views, i.e., attributes that do not appear in the head of views, the assumption of independence between constraints is guaranteed and the degrees can be aggregated with the product operation. Indeed, if a view covers a query subgoal as well as an interval constraint over one of its existential variables, it must also cover any query subgoal involving this variable [15]. Consequently, in a same rewriting, it is not possible to have more than one view that restricts the value domain of an existential variable. The only case where the independence assumption may be violated concerns *distinguished* variables in views, i.e., attributes that appear in the head of views. Indeed, several views occurring in a given rewriting may involve interval constraints on a same distinguished variable. In such a case, the overall degree cannot be computed as an aggregation of partial degrees, but must be based on the intersection of the interval constraints. The degree attached to a distinguished variable is either 0 or 1, depending on whether the new interval is disjoint or not from that of Q. If it is 0, the rewriting is dismissed, whereas if it is 1, the constraint from Q is added to the rewriting.

As an example, let us now consider a query Q which aims at retrieving *codes* of *OMSParcels* whose *surface* is in [22, 35] hectares and located in a city with a wastewater treatment plant ($STEP$) whose *capacity* is in [1200, 2300], and two views:

$V_1(code,\ citycode) :-$
$\quad OMSParcels(code,\ surface,\ citycode),\ surface \in [18,\ 38]$
$V_2(citycode) :- STEP(citycode,\ capacity),\ capacity \in [1000, 2400].$

Let $Q_1(code) :- V_1(code, citycode), V_2(citycode)$ be a tolerant rewriting of Q. The degree α_1 attached to the tolerant rewriting V_1 of the first subgoal of Q is computed from the interval constraints on surface and it equals $\frac{|[22,35]|}{|[18,38]|} = \frac{13}{20}$ while the degree α_2 attached to the tolerant rewriting V_2 of the second subgoal of Q is computed from the interval constraints on capacity and it equals $\frac{|[1200,2300]|}{|[1000,2400]|} = \frac{11}{14}$. Therefore, $Q_1(code)$ gets the degree $\alpha_1 * \alpha_2 = 0.51$ and $Q_1 \sqsubseteq_{0.51} Q$. The tuples issued from Q_1 have a probability over 50% to satisfy the constraints involved in Q.

As can be seen, the rewriting mechanism proposed is not based on the notion of *certain answers* anymore, but rather on that of *probable answers*. The idea of computing probable answers is not new since it has been used for open integration contexts [4, 19]. The nature of the probability degree associated with the answers returned in our approach is different from that considered in these works, though. Indeed, in [4] it indicates the probability of *existence* of the answers produced, while in [19] it ensues from the uncertainty pervading the mappings between several data sources.

Since approximate mappings between queries and views are obviously more frequent than strict mappings, the number of possible tolerant rewritings may be huge. To cope with this situation, we propose an algorithm which generates only the top-k rewritings to a given query. This algorithm rests on the principle of a well-known regular query rewriting algorithm, namely `Minicon` [15]. It slightly modifies the first step of the `Minicon` by associating a degree with any view able to rewrite a subgoal of the query. Subsequently, as the second step of the `Minicon` amounts to the computation of the exact covers of a hypergraph (i.e., those of the subgoals of a query), our algorithm exploits an efficient structure to implement it, called *dancing links* [9], in order to generate only the k best rewritings.

5. EXPERIMENTS

In a decision-making context, the different protagonists of the agricultural domain need to exchange data. Each protagonist should be able to obtain information according to his specific requirements: a farmer may need information for choosing the place of his crops, a public authority needs to monitor the environmental impacts of agricultural activities, etc. Due to the generalization of the traceability of the agricultural practices, each protagonist produces more and more data. Indeed, the amount of data involved is huge. For example, in France one counts more than 500,000 farms and thousands of agencies or services that take part in the agricultural activity. The need for exchanging and querying this information is crucial in this application. Experts in the field of agriculture in France participated in the project. They helped propose a coherent exchange scenario by identifying the possible protagonists and their data needs. Even if we did not deploy our system at a national level in France, the goal was to provide a scenario sufficiently realistic to be implemented. Numerous data sources have been collected for the project. We have conducted experiments for integrating 300 data sources issued from different French institutions and farms. More precisely, data sources are extractions of databases or Excel files. Each extraction is in fact a view (e.g., a table). The first challenging issue is to build the mediated schema over the different data sources. Many concepts are common to different sources (environmental objects, crops, farms, etc) but the expertise of each source is specific. Designing the mediated schema required finding the common concepts between the sources. Manually finding the correspondences between sources is an hard task for experts. We used MatchPlanner, the automatic schema matching tool presented in Section 2. The list of correspondences that it provided helped highlight the concepts repeated in several sources i.e. the potential relations of the mediated schema. The experts (in)validated this list manually by removing the false positives and by adding new correspondences. This process allowed to discover forty common concepts (each one corresponds to a table of the mediated schema). Experts highlighted that one can save a significant amount of time by using the schema matching tool.

Then, the mappings between the mediated schema and the data sources were described manually following an LAV approach. Each mapping was modelled as a Datalog clause; several hundreds of Datalog rules were thus written. Here an example of such a mapping:

```
wastewaterPlantRegion1(wwpIdSandre,wwpName,
     managerName,managerId):-
     manager(_,managerId,_,managerName,_,_)
     wastewaterPlant(_,wwpIdSandre,
                     wwpName,_,_,_,
                     managerIdSiret).
```

`wastewaterPlantRegion1` is a source gathering the list of wastewater plants that belong to the geographical area 1. In the body of the rule, `manager` and `wastewaterPlant` are tables in the mediated schema. Five variables are involved in this mapping:

1. wwpIdSandre: id of the wastewater plant expressed in format named Sandre (French format),
2. wwpName: tname of the wastewater plant,
3. managerName: name of the main manager of a wastewater plant,
4. managerId: id of the manager of a wastewater plant,
5. managerIdSiret: id of the managers expressed in a specific format called Siret (French format).

So as to test the flexible rewriting technique, the experts expressed certain attributes values as intervals of values. Note that for many queries there are only rewritings with a degree less than 1. Therefore, in many cases a classical approach does not compute any rewriting.

The approach has been implemented in a prototype using Java SE 5, and PostgreSQL 8.3 databases. Twenty queries provided by real users were processed. The prototype computes the best rewritings

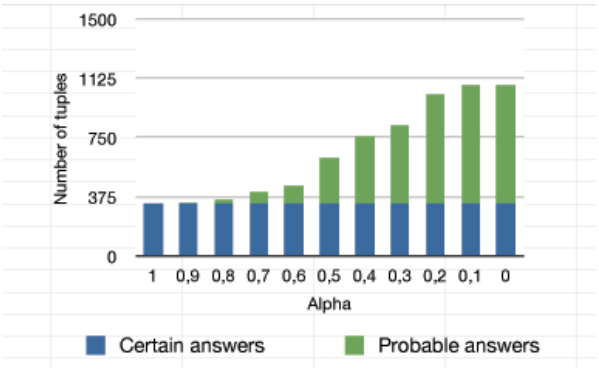


Figure 2: Impact of α_{min}

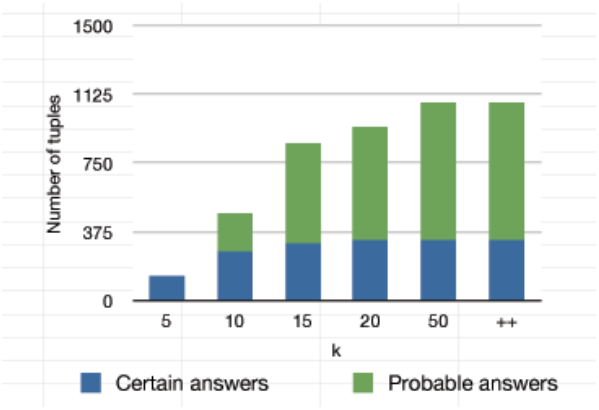


Figure 3: Impact of k

associated with a given query: either the top-k ones (k being an integer) or those whose associated degree is over a given threshold $\alpha_{min} \in]0, 1]$. The first experimentation (cf. Fig. 2) shows the evolution of the *average* number of answers, w.r.t. the number of certain anwers computed by the `MiniCon` algorithm, when the threshold α_{min} changes. Here the parameter k is fixed to infinity. Notice that the number of certain answers remains constant. Indeed, all exact rewritings are computed for any value of α_{min}. On the other hand, the number of probable answers decreases when the threshold increases. When α_{min} equals 1, only certain answers are provided. In the second experimentation (cf. Fig. 3), the threshold is set to 0 but k, the number of best rewritings sought for, must be specified. When k decreases, the average number of probable answers decreases too, which shows that only the best rewritings are generated. When k is less or equal than 10, the number of certain answers decreases too. Indeed, for some queries, a number of 10 rewritings is not enough to compute all the certain answers. The prototype shows very good performances: the execution time is about 2 seconds in the worst case, and the performances of the approach are comparable to that of the classical `MiniCon` algorithm (in the worst case, the overhead is only 10%). See [8] for more detail.

6. CONCLUSION

The main objective of the FORUM project was to develop new methods and techniques for flexible data integration. This project has led to significant advances in the field. The results were published in international journals and conferences. Original and efficient tools have been developed and experimented in an environmental application.

Finally, the project has enabled the development of collaborations between partners. For example, a collaboration between LIMOS and IRISA teams led to the development of a novel approach for flexible query rewriting. A collaboration between LIRMM and IRISA will begin soon on schema integration using fuzzy set and possibility theories to capture uncertainty resulting from the use of similarity measures.

7. ACKNOWLEDGMENTS

Work partially funded by the ANR Research Grant ANR-05-MMSA-0007. We would like to thank the reviewers for their very fruitful comments to improve the paper.

8. ADDITIONAL AUTHORS

Stephan Bernard (CEMAGREF, stephan.bernard@cemagref.fr), Pierre Colomb (LIMOS, colomb@isima.fr), Remi Coletta (LIRMM, coletta@lirmm.fr), Emmanuel Coquery (LIRIS, emmanuel.coquery@liris.cnrs.fr), Fabien de Marchi (LIRIS, fabien.demarchi@liris.cnrs.fr), Fabien Duchateau (LIRMM, duchatea@lirmm.fr), Mohand-Saïd Hacid (LIRIS, mohand-said.hacid@liris.cnrs.fr), Allel Hadjali (IRISA, hadjali@enssat.fr) and Mathieu Roche (LIRMM, mroche@lirmm.fr)

9. REFERENCES

[1] F. N. Afrati, C. Li, and P. Mitra. Answering queries using views with arithmetic comparisons. In *PODS*, pages 209–220, 2002.

[2] B.Alexe, M. Hernández, L.Popa, and W.C.Tan. Mapmerge: Correlating independent schema mappings. *PVLDB*, 3(1):81–92, 2010.

[3] P. A. Bernstein, T. J. Green, S. Melnik, and A. Nash. Implementing mapping composition. *VLDB J.*, 17(2):333–353, 2008.

[4] N. Dalvi and D. Suciu. Answering queries from statistics and probabilistic views. In *Proc. of VLDB 2005*, pages 805–816. VLDB Endowment, 2005.

[5] H. H. Do, S. Melnik, and E. Rahm. Comparison of schema matching evaluations. In *IWWD*, 2003.

[6] F. Duchateau, Z. Bellahsene, and R. Coletta. A flexible approach for planning schema matching algorithms. In *OTM Conferences*, pages 249–264, 2008.

[7] F. Duchateau, Z. Bellahsene, and M. Roche. A context-based measure for discovering approximate semantic matching between schema elements. In *RCIS*, 2007.

[8] H. Jaudoin, P. Colomb, and O. Pivert. Ranking approximate query rewritings based on views. In *Proc. of the 8th International Conference on Flexible Query Answering Systems (FQAS'09)*, pages 13–24, Roskilde, Denmark, 2009.

[9] D. Knuth. Dancing links. *Arxiv preprint cs.DS/0011047*, 2000.

[10] P. G. Kolaitis. Schema mappings, data exchange, and metadata management. In *PODS '05: Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 61–75, New York, NY, USA, 2005. ACM.

[11] C. Li and C. Clifton. Semantic integration in heterogeneous databases using neural networks. In *VLDB*, 1994.

[12] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *VLDB*, pages 49–58, 2001.

[13] T. Milo and S. Zohar. Using schema matching to simplify heterogeneous data translation. In *VLDB*, pages 122–133, 1998.

[14] M. Murata, D. Lee, M. Mani, and K. Kawaguchi. Taxonomy of XML schema languages using formal language theory. *ACM Trans. Internet Technol.*, 5(4):660–704, 2005.

[15] R. Pottinger and A. Y. Levy. A scalable algorithm for answering queries using views. In *VLDB*, pages 484–495, San Francisco, CA, USA, 2000.

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

[17] R.Fagin, P. Kolaitis, R.Miller, and L.Popa. Data exchange:semantic and query answering. In *Proc. ICDT 2003*, 2003.

[18] K. Saleem, Z. Bellahsene, and E. Hunt. PORSCHE: Performance ORiented SCHEma mediation. *Information Systems - Elsevier*, 33(7-8):637–657, 2008.

[19] A. D. Sarma, X. Dong, and A. Halevy. Bootstrapping pay-as-you-go data integration systems. In *Proc. of SIGMOD 2008*, pages 861–874. ACM, 2008.

[20] Secondstring. http://secondstring.sourceforge.net/, 2007.

[21] P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. *J. Data Semantics IV*, pages 146–171, 2005.

[22] Y. Velegrakis, R. Miller, and L. Popa. Preserving mapping consistency under schema changes. *The VLDB Journal*, 13(3):274–293, 2004.

[23] Wordnet. http://wordnet.princeton.edu, 2007.

VSkyline: Vectorization for Efficient Skyline Computation

Sung-Ryoung Cho† Jongwuk Lee‡ Seung-Won Hwang‡
Hwansoo Han† Sang-Won Lee†

†School of Information & Communications Engineering
Sungkyunkwan University
Suwon, 440-746, Korea
{applys,hhan,swlee}@skku.edu

‡Department of Computer Science and Engineering
Pohang University of Science and Technology
Pohang, 790-784, Korea
{julee,swhwang}@postech.edu

ABSTRACT

A dominance test, which decides the *dominance* relationship between tuples, is a core operation in skyline computation. Optimizing dominance tests can thus improve the performance of all existing skyline algorithms. Towards this goal, this paper proposes a *vectorization* of dominance tests in *SIMD architectures*. Specifically, our vectorization can perform the dominance test of multiple consecutive dimensions in parallel, thereby achieving a speedup of *SIMD parallelism degree* in theory. However, achieving such performance gain is non-trivial due to complex control dependencies within the dominance test. To address this problem, we devise an efficient vectorization, called VSkyline, which performs the dominance test with SIMD instructions by determining *incomparability* in a block of four dimensional values. Experimental results using a performance monitor show that VSkyline considerably reduces the numbers of both executed instructions and branch mispredictions.

1. INTRODUCTION

For the past decade, skyline queries [2, 3, 4, 5, 6, 7, 11] have gained considerable attention for helping multi-criteria decision-making processes in a large-scale data set of tuples. Because skyline computation depends heavily on tuple-wise comparisons to check the *dominance* relationship between tuples, called *dominance tests*, existing skyline algorithms tend to focus on avoiding unnecessary dominance tests either by pruning non-skyline candidates early or by partitioning an entire data set into multiple subsets.

Despite such optimizations, a huge number of dominance tests is still inevitable in skyline computation. In particular, as dimensionality increases, the number of dominance tests increases exponentially. Since the dominance test is a core operation in skyline computation, we aim to optimize the dominance test itself. Specifically, when the values of tuples are scanned sequentially for the dominance test, there exists an opportunity to take advantage of data-level parallelism in *SIMD* (*single instruction, multiple data*) architectures. Although SIMD architectures are available in almost all modern CPUs, to the best of our knowledge, no skyline algorithm makes use of SIMD architectures to achieve efficient dominance tests.

In this paper, we propose a *vectorization* of dominance tests, which can determine dominance relationship efficiently, by performing comparison for multiple consecutive dimensions in parallel. Given a SIMD parallelism degree, *e.g.*, currently four, we can boost the performance by up to four folds in theory. However, achieving such performance gain is non-trivial for skyline computations. In general, a dominance test on multiple dimensions should be performed in a sequential order for each dimension, as the result of dominance test at a specific dimension is dependent on the results of preceding dimensions and the result for multiple dimensions is thus decided at the last dimension. For these dependencies, SIMD comparison of multiple dimensions requires a series of conditional branches to inspect the comparison result at each dimension with consideration of the preceding dominance test results. As the performance of SIMD architectures can be greatly hindered by complex conditional branches [9], a naive-vectorization thus performs even worse than a non-vectorization version.

To address this problem, we devise an efficient vectorization, called VSkyline, which performs the dominance test with SIMD instructions by determining *incomparability* in a block of four dimen-

sional values. Our performance evaluation using a performance monitor shows that VSkyline considerably reduces the numbers of both instructions including branches and branch mispredictions. We also stress that this speedup is orthogonal to existing parallelization efforts, which suggests that the existing algorithms can benefit from our finding as well.

The rest of this paper is organized as follows. Section 2 briefly reviews skyline computation and SIMD technology. Section 3 proposes an efficient vectorized skyline algorithm VSkyline, and Section 4 then validates performance evaluation of VSkyline in extensive synthetic data sets. Finally, Section 5 concludes our paper.

2. BACKGROUND

2.1 Skyline Queries

We first introduce basic notations to address skyline query problem. Let $\mathcal{D}$ be a finite d-dimensional space, *i.e.*, $\{D_1, \ldots, D_d\}$, where each dimension has a domain $dom(D_i)$ of non-negative real number $\mathbb{R}^+$. Let $\mathcal{S}$ be a set of finite n tuples as a subset of $dom(\mathcal{D})$. A tuple p in $\mathcal{S}$ is represented as $p = (p_1, \ldots, p_d)$, where $\forall i \in [1, d] : p_i \in dom(D_i)$.

Based on these notations, we formally states some notions commonly used in the skyline literature [2, 3, 4, 5, 6, 7, 11]. Throughout this paper, we consistently use *max* operator for skyline queries. Specifically, given two tuples p and q, p *dominates* q on $\mathcal{D}$ if $\forall\ i \in [1, d] : p_i \geq q_i$ and $\exists\ j \in [1, d] : p_j > q_j$, denoted as $p \succ q$. Also, it is said p and q are *incomparable* if they do not dominate each other, denoted as $p \sim q$.

Given $\mathcal{S}$, a skyline query on $\mathcal{D}$ thus returns a subset of tuples, or *skyline*, that are no worse than, or not dominated by, any other tuples in $\mathcal{S}$, *i.e.*, $\{p \in \mathcal{S} | \nexists q \in \mathcal{S} : q \succ p\}$. A tuple in skyline is called a *skyline tuple*.

The skyline computation depends heavily on dominance tests between tuples. In the worst case, a naive skyline algorithm exhaustively performs dominance tests for all possible $n(n-1)/2$ tuple pairs, incurring quadratic cost.

To address this problem, existing skyline algorithms aimed to reduce unnecessary dominance tests. Specifically, they could be classified into two categories: (1) *sorting-based* algorithms such as BNL [2], Index [11], SFS [3], and LESS [4] focused on optimizing tuple ordering to prune more dominated tuples early on; (2) *partitioning-based* algorithms such as NN [5], BBS[7], ZSearch [6], and OSPS [13] focused on dividing an entire dataset into multiple subsets to exploit "region-level" optimization.

On the other hand, due to the CPU-intensive property of dominance tests [8, 12], skyline computation can take advantage of *parallelism* as two forms – (1) *thread-level parallelism* and (2) *data-level parallelism*.

All existing parallel skyline algorithms fall into the category of *thread-level parallelism*, distributing partitioned subsets into independent threads, and then merging local skylines into global skyline. Specifically, Vlachou et al. [12] developed angle-based data partitioning to balance the computation overhead between threads in data distribution. Recently, Park et al. [8] proposed PSkyline, which optimizes merge processing to enhance thread utilization, by computing multiple dominance tests in parallel.

To the best of our knowledge, no existing skyline algorithm has attempted data-level parallelism [9], performing the same operation on multiple data simultaneously. We propose a *vectorization* of dominance tests to achieve a theoretical speedup of SIMD parallelism degree. We stress that this speedup is orthogonal to thread-level parallelism, and skyline algorithms using thread-level parallelism can thus exploit data-level parallelism as well.

2.2 SIMD Technology

Almost every modern CPU supports SIMD instructions. For instance, the x86 microprocessor provides hundreds of MMX (multi-media extensions) and SSE (streaming SIMD extensions) instructions. A SIMD instruction operates on vectors of data. Therefore, one obvious performance benefit of using SIMD instructions is to process multiple elements at a time, and we can expect a speedup of SIMD parallelism degree, denoted as V, in theory.

SIMD operations can work best when handling a great deal of identically structured data, *e.g.*, arrays in *for* loop, for data-level parallelism. For example, many database algorithms such as sequential scans, scalar aggregations, index traversals, and joins perform repetitive operations on an array of tuples (and each tuple in turn is an array of columns). Zhou and Ross [14] showed how the inner loop of those operations can benefit from using SIMD instructions. We call this transformation *vectorization* (or *SIMDization*).

Along the same line, skyline computation is also a good candidate for vectorization. It requires numerous tuple-wise dominance tests in a large set of tuples, where a dominance test sequentially accesses two tuples in array structures.

However, it is a non-trivial task to successfully

vectorize dominance tests. It is well known that SIMD is at its weakest in *control* statements [9], and unfortunately complex conditional branches are innate in dominance tests. Furthermore, conditional branches are problematic in modern pipelined architectures because of serious branch misprediction penalty from instruction pipeline flush and other bookkeeping overheads ensuring operational consistency [1].

For this reason, if we can reduce or eliminate conditional branch instructions during vectorization, it might provide potentially larger performance gain. For example, Zhou and Ross [14] showed that they can obtain considerable performance enhancement by eliminating conditional branch instructions while vectorizing database operations, thus avoiding the branch misprediction penalty. However, the vectorization techniques are relatively simple because the result on each element is independent from other elements.

In clear contrast, in the dominance test, the result of *incomparability* at a specific dimension is also dependent on the results at previous dimensions. This property of *conditional dependency* can make a naive vectorization of dominance tests even poorer than a non-vectorized version. The obvious benefit of vectorization with SIMD instructions is the decrease of loop iterations, resulting in less number of executed instructions. A naive vectorization, however, can require more extra instructions to manipulate the values in SIMD registers, which negates the benefit of SIMD vectorization. To address this problem, we can limit the number extra instructions and make complex conditional control flow more predictable. As a result, the total number of executed instructions and mispredicted branches are reduced, which ultimately improves the performance. The next section discusses an efficient vectorization of dominance tests.

3. SKYLINE COMPUTATION

This section first overviews a non-vectorized dominance test in PSkyline, and shows that a naive vectorization is inefficient to handle complex conditions resulting from SIMD comparisons. We then propose an efficient vectorization for dominance tests, called VSkyline, which optimizes the complex condition tests in the dominance test. We apply our vectorization technique to PSkyline. Since data-level parallelism in VSkyline is orthogonal to thread-level parallelism in PSkyline, we can further improve the performance for PSkyline by using our vectorization technique.

Algorithm 1 PSkyline dominance test

```
1:  Dom ← Incomparable; i ← 1;
2:  while i ≤ d do
3:      if p_i < q_i then
4:          if Dom = Left then
5:              return Incomparable;
6:          end if
7:          Dom ← Right;
8:      else if p_i > q_i then
9:          if Dom = Right then
10:             return Incomparable;
11:         end if
12:         Dom ← Left;
13:     end if
14:     i ← i + 1;
15: end while
16: return Dom;
```

3.1 PSkyline Overview

PSkyline [8] extends skyline computation leveraging thread-level parallelism, by dividing an entire dataset into multiple subsets and merging local skylines from each thread. Specifically, the *map* process is to distribute the dataset to threads and compute each local skyline. Then, *reduce* process is to merge local skylines into a global skyline in parallel. A key contribution of PSkyline is to optimize thread utilization for both *pmap* (*map* process) and *pmerge* (*reduce* process). Conceptually, *pmap*, among the tuples assigned to each thread, prunes out some of non-skyline tuples that cannot be contained in a final skyline by using *partial sorting*. Also, *pmerge* aggregates the local results, and returns the final skyline by maximizing thread-level parallelism. These two modules, however, rely on the classic pair-wise dominance test.

To illustrate, Algorithm 1 describes the dominance test in PSkyline as well as most of the existing skyline algorithms. Given two tuples p and q, p_i and q_i represent the i-th dimensional values. Typically, the dominance test starts with comparing p_1 and q_1 values, iterating the loop until the dominance relationship is decided.

Specifically, in an iteration of the loop, Dom is set to either $Left$ or $Right$. If the result of the current iteration is different from that of the previous iteration, $Incomparable$ is returned. That is, even in the best case, the result is obtained at the second iteration. Returning $Incomparable$ means two tuples are incomparable. In such case, any further comparison on the rest of the dimensions is unnecessary.

We could see that a single dominance test includes quite a few conditional branches, which are particularly unpredictable. Since comparison results of dimensional values can be any, the condi-

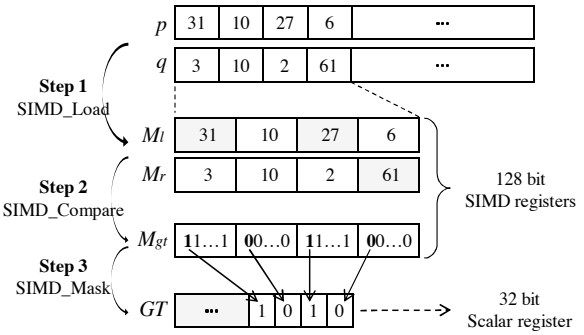


Figure 1: SIMD operations

tional statements (lines 3-13) in Algorithm 1 can flow any direction. This actually makes conditional branches in the dominance test unpredictable. This unique characteristic of the dominance test suggests that modern CPUs with deep pipelines will suffer from branch mispredictions, which significantly degrade the performances. Indeed, it is commonly believed that even 5% branch misprediction can degrade performance by as much as 20∼30% in modern CPUs [14]. We also observe similar symptoms from PSkyline. The mispredicted branches account for about 15∼20% of the CPU cycles for PSkyline.

3.2 Vectorization and Packed Conditions

Typical dominance tests proceed sequentially one dimension by one dimension as in Algorithm 1. To vectorize dominance tests, we need to compare four dimensions simultaneously with SIMD instructions, where SIMD parallelism degree is four. Thus, the comparison results across four dimensions are stored in a SIMD register. Since the result of the dominance test has a dependence on the comparison result of previous dimensions, interpreting the SIMD comparison result requires extra instructions to manipulate SIMD registers.

Figure 1 illustrates the steps of the vectorized dominance test over four dimensions. SIMD_Load, SIMD_Compare, and SIMD_Mask operations are performed in order and four bits are computed as a final summary of SIMD comparison. SIMD registers, M_l, M_r, and M_{gt} are 128 bit long each. A general register, GT is 32 bit long. Each SIMD register can keep four elements, each of which is 32 bit long type (e.g. `int` or `float`). SIMD register M_{gt} contains the result of *greater-than* comparison. If element-wise comparison is true, all 32 bits for that element are set to 1s. Otherwise, all 32 bits are set to 0s. SIMD_Mask summarizes the SIMD comparison result to four bits by taking the most significant bit of each element. Those four bits are stored in the lower four bits of GT register. Throughout this paper, we call those four bits *significant bits*.

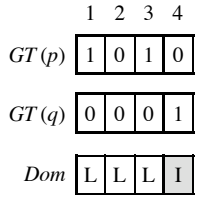


Figure 2: A dominance test in NaiveSIMD

Interpreting the significant bits for the dominance test can be implemented in a naive way (NaiveSIMD) or an optimized way (VSkyline). To evaluate incomparability between two tuples, we need to examine the result of each dimension comparison, and determine the dominance result with consideration of the previous dimensions. Because comparison results of four dimensions are packed into significant bits, the conditions to decide the dominance result are also packed into those significant bits. We call this situation *packed conditions*. In the next section, we discuss why a naive interpretation of packed conditions causes performance loss, and propose an optimized interpretation for packed conditions.

3.3 Vectorized Dominance Test

3.3.1 NaiveSIMD

Figure 2 illustrates a naive vectorization of the dominance test for two tuples shown in Figure 1. Bits in $GT(p)$ are set when p is greater than q and bits in $GT(q)$ are set when q is greater than p. *Dom* shows the intermediate result of the dominance test up to the corresponding dimension. L, R, and I represent *left-dominate*, *right-dominate*, and *incomparable*, respectively. NaiveSIMD uses two comparison results and computes dominance results with the same control logic as Algorithm 1. Examining the comparison results one dimension by one dimension, NaiveSIMD determines the dominance result by using the same control logic in Algorithm 1 except that it performs a SIMD comparison over four dimensions.

Up to third dimension, the intermediate result in *Dom* is *left-dominating (L)*. At the forth dimension, its comparison result is inconsistent with the previous dimension. Thus, the final dominance test for p and q becomes *incomparable (I)*. NaiveSIMD often requires many extra instructions, such as shift, bitwise logical operations, to manipulate the results stored in significant bits. If we count the number of executed instructions for the case depicted in Figure 2, NaiveSIMD executes 1.5x more instructions than PSkyline. Moreover, PSkyline suffers from a

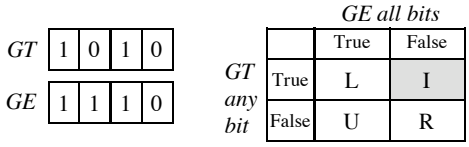


Figure 3: A dominance test in VSkyline

sizable number of mispredicted branches, which accounts for 15∼20% of execution cycles on various inputs. NaiveSIMD shares the same characteristics. Thus, NaiveSIMD performs worse than PSkyline.

To maximize the benefit from SIMD vectorization, we need to devise a new dominance test logic, which can examine comparison results as a whole, not by individual dimensions. In the next section, we introduce VSkyline, which can effectively handle packed conditions with limited number of extra instructions and far less number of branch mispredictions.

3.3.2 VSkyline

Figure 3 illustrates how VSkyline performs the dominance test by taking the comparison results of four dimensions as a whole. Similar to NaiveSIMD, we need two comparison results, but slightly different comparisons, GT (*greater-than*) and GE (*greater-than-or-equal*). Then, we decide the dominance result over four dimensions at once by using the truth table shown in the right side of Figure 3. NaiveSIMD examines one dimension by one dimension, even after parallelizing compares with SIMD instructions. On the contrary, our VSkyline examines all four dimensions as a whole.

As shown in the truth table, we examine two conditions. If any bit in GT is set to 1, the *GT-any-bit* becomes true. Otherwise, false. If all bits in GE are set to 1s, the *GE-all-bits* becomes true. Otherwise, false. Depending on two conditions, we can decide the dominance result up to four dimensions at once. L, R, I, and U represent *left-dominating*, *right-dominating*, *incomparable* and *undecided*, respectively. *Undecided* means two tuples have the same values for all dimensions so far and further test on the rest of the dimensions are required to decide the dominance result. Using the same example in Figure 1, its dominance result in VSkyline is *incomparable*, which is the same result shown in Figure 2.

In general, the dominance test in VSkyline categorizes packed conditions into the following four cases. Each case corresponds to a dominance result of four dimensions, which is shown in the truth table of Figure 3.

Algorithm 2 VSkyline dominance test

```
 1: Dom ← Incomparable; i ← 1;
 2: while i ≤ d do
 3:     M_l, M_r ← SIMD_Load(p[i, i+V−1], q[i, i+V−1]);
 4:     M_gt, M_ge ← SIMD_Compare(M_l, M_r);
 5:     GT, GE ← SIMD_Mask(M_gt, M_ge);
 6:     if GE is set in all significant bits then
 7:         if GT is set in any significant bit then //Case 1
 8:             if Dom = Right then
 9:                 return Incomparable;
10:             end if
11:             Dom ← Left;
12:         end if
13:         //Undecided: Case 4
14:     else if GT is set in any significant bit then //Case3
15:         return Incomparable;
16:     else //Case 2
17:         if Dom = Left then
18:             return Incomparable;
19:         end if
20:         Dom ← Right;
21:     end if
22:     i ← i + V;
23: end while
24: return Dom;
```

Case 1 [**Left_dominating**] if $\forall i \in [1, V]$: $p_i \geq q_i \wedge \exists j \in [1, V]$: $p_j > q_j$, then $p \succ q$ holds.

Case 2 [**Right_dominating**] if $\forall i \in [1, V]$: $p_i \leq q_i \wedge \exists j \in [1, V]$: $p_j < q_j$, then $q \succ p$ holds.

Case 3 [**Incomparable**] if $\exists i \in [1, V]$: $p_i > q_i \wedge \exists j \in [1, V]$: $p_j < q_j$, then $p \sim q$ holds.

Case 4 [**Undecided**] if $\forall i \in [1, V]$: $p_i = q_i$, then the dominance relationship is not decided yet.

VSkyline executes far less instructions. As we need not to iterate each dimension to interpret the packed conditions, lots of extra instructions to handle loops and SIMD register values are actually eliminated. If NaiveSIMD can determine dominance result early for all data, it probably performs well enough to compete with VSkyline. However, skyline algorithms frequently handle large datasets and the average number of dimensions to examine is more than just one or two. In such cases, VSkyline can outperform NaiveSIMD for most of cases. Even compared to PSkyline, we execute less number of instructions, as we handle four dimensions in one iteration. According to our experiments, executed instructions are reduced by 28∼45% for various inputs, which is translated to the CPU cycle reductions by 30∼60%.

In addition, handling packed conditions from SIMD comparisons produces more predictable control structures. Since we executes less number of branches and dominance results are decided at once for multiple dimensions, hardware branch predictors can

work more favorably in our algorithm. Meanwhile, both PSkyline and NaiveSIMD process each dimension one by one, which results in more diverse outcomes for the same branch instructions, thus making hardware branch predictors difficult to predict their branch directions. Performance gap between the two and VSkyline is largely due to far less mispredicted branches. According to our experiments, the number of mispredicted branches are reduced by 53∼97% for various inputs, which is translated to the CPU cycle reductions by 10∼20%.

Algorithm 2 shows the vectorized version of the dominance test used in VSkyline. Three main operations such as SIMD_Load, SIMD_Compare, and SIMD_Mask is illustrated in Figure 1 with slight modifications. Specifically, SIMD_Load read four dimensions from each tuple to compare, which is the same. Modifications are applied to SIMD_Compare and SIMD_Mask. SIMD_Compare generates two results in M_{gt} and M_{ge} with *greater-than* and *greater-than-or-equal* comparisons, respectively. Similarly, SIMD_Mask summarizes two comparison results of M_{gt} and M_{ge} into GT and GE, respectively. Conditional control flows which test two conditions, *GE-all-bits* and *GT-any-bit*, lead to four cases – *left-dominating*, *right-dominating*, *incomparable* and *undecided* as specified in the algorithm.

In summary, two performance boosters of VSkyline are as follows: 1) reduction in number of executed instructions due to SIMD vectorization of dimension value load and compare, and 2) reduction in number of mispredicted branches due to simplified control. In the next section, we show experimental results for those performance boosters.

4. EXPERIMENTAL EVALUATION

This section reports our evaluations results by comparing our proposed algorithm VSkyline with PSkyline. All algorithms were implemented in C language, and all experiments were conducted on a Linux server with a 2.6.18 kernel. The server was also equipped with an Intel i7-860 quad core processor running at 2.80GHz and 6GB main memory. To analyze the performance, we used Intel VTune performance analyzer [10].

We used synthetic datasets with the same experimental settings in PSkyline [8]. Specifically, we generated two different data sets according to uniform and anti-correlated distributions. To measure scalability, we varied the dimensionality from 4 to 24 with an increment of two and the cardinality from $50K$ to $400K$ with a double increment in numbers. Similarly, we varied the number of threads from 1, 4, then 8. As SIMD parallelism degree is four, we executed the maximum multiple of four dimensions with SIMD instructions and the remaining dimensions with non-SIMD instructions. Again, we clarify that our key contribution is not to reduce the number of tuple-to-tuple tests, but the cost of each test by using SIMD vectorization. As a result, we observed that VSkyline, performing the same number of tests at a lesser cost, outperforms PSkyline in all cardinality from $50K$ to $400K$. To present multiple aspects of the performance, we used the results for two cardinalities of $100K$ and $200K$.

4.1 Effects of Vectorization

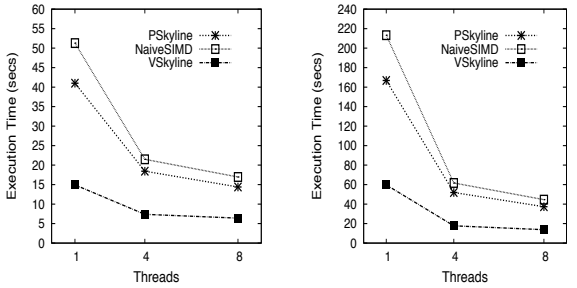
Figure 4 depicts the effects of vectorization in synthetic datasets. VSkyline outperforms PSkyline by up to three folds. Observe that, in Figure 4, NaiveSIMD consistently underperforms PSkyline over varying number of threads. The same observation holds for different cardinality and dimensionality. A possible explanation to such performance degradation is the naive approach in handling packed conditions, which cancels out the advantages of SIMD vectorized loads and compares.

Specifically, Table 1 compares three skyline algorithms, PSkyline, NaiveSIMD, and VSkyline, with two datasets on different number of threads. Uniform and anti-correlated datasets, each of which has 12 dimensions and $200K$ data points, are used for our detailed performance analysis. The numbers for 4, 8 threads display the measured numbers on one thread, as numbers on other threads are very similar. Compared to PSkyline, the numbers of executed instructions (NUM_INST) and branches (NUM_BR) are reduced by 30∼35% and 52∼63%, respectively. Due to the reduced number of iterations in dominance tests, VSkyline executes less instructions including branches than PSkyline. Meanwhile, NaiveSIMD examines packed conditions one dimension by one dimension, which makes it execute more instructions even than PSkyline. Since extra instructions to manipulate SIMD results such as shift and bit mask instructions are required in NaiveSIMD, it needs to execute more instructions.

One of good characteristics in VSkyline is the reduced number of mispredicted branches. As it processes packed conditions as a whole with two SIMD comparisons, control flow becomes more predictable than others. Thus, its mispredicted branch ratios (BR_MISP) become far smaller than the other algorithms, implying the performance gain of VSkyline. Specifically, compared to PSkyline, the execution cycle (CPU_CLK) of VSkyline is reduced by 55∼65%. We also measured detailed performance counts with various dimensionality and cardinality and observed

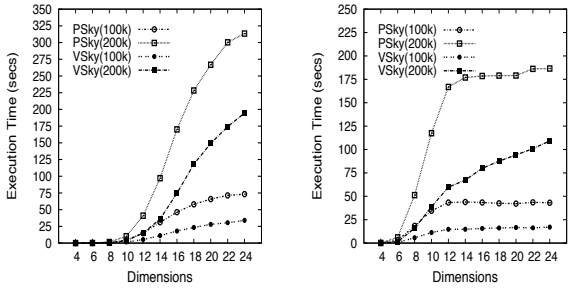
Table 1: Performance comparison of skyline algorithms (12d, 200K)

threads	algorithms	Uniform CPU_CLK (cycle)	Uniform NUM_INST (million)	Uniform NUM_BR (million)	Uniform BR_MISP (%)	Anti-correlated CPU_CLK (cycle)	Anti-correlated NUM_INST (million)	Anti-correlated NUM_BR (million)	Anti-correlated BR_MISP (%)
1	PSkyline	137,104 M	121,850	44,803	6.66	559,313 M	497,099	181,660	6.51
	NaiveSIMD	171,539 M	224,117	47,410	6.52	711,399 M	928,903	195,775	6.40
	VSkyline	49,469 M	79,296	16,919	2.91	199,643 M	330,107	68,105	2.21
4	PSkyline	54,127 M	43,613	16,139	7.74	151,849 M	130,304	47,843	6.55
	NaiveSIMD	62,898 M	78,650	17,951	7.04	180,548 M	240,708	55,013	6.04
	VSkyline	21,614 M	29,520	6,977	5.30	50,862 M	90,170	20,371	2.38
8	PSkyline	40,955 M	26,170	7,890	9.71	108,988 M	74,628	22,247	7.90
	NaiveSIMD	49,073 M	46,246	9,029	8.81	128,822 M	132,013	25,625	7.14
	VSkyline	18,350 M	17,710	3,752	6.86	39,456 M	49,159	10,087	2.60



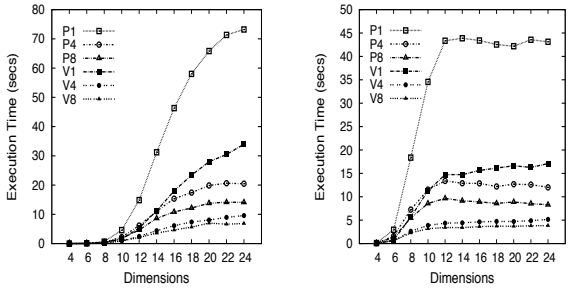
(a) Uniform (b) Anti-correlated

Figure 4: Performance of algorithms (12d, 200K)



(a) Uniform (b) Anti-correlated

Figure 5: Scalability on single-core (100K, 200K)



(a) Uniform (b) Anti-correlated

Figure 6: Scalability on multi-core (100K)

similar trends to the case in Table 1. The number of mispredicted branches, not only the number of instructions are reduced, improving the performance of VSkyline.

4.2 Effects of Dimensionality

Theoretical maximal speedup that can be obtained from vectorization is V-fold, where V indicates the degree of data-level parallelism. In our experiment, V is four as our SIMD instructions handle four data values. SIMD vectorization, however, requires extra instructions to manipulate the result stored in SIMD registers. Thus, actual speedup cannot reach the theoretical optimum, as VSkyline incurs the same overheads. In addition, it has additional computational overheads of deciding how to branch, after comparisons, which further hinders from achieving the optimal speedup. Figure 5 reports evaluation results of PSkyline and VSkyline in uniform and anti-correlated datasets over varying cardinality (100K and 200K). We can observe that, VSkyline consistently outperforms PSkyline, though the speedup varies over different parameter settings.

Figure 6 extends these evaluations, varying the number of threads. The solid markers indicate the results of VSkyline, denoted by Vx, when x indicates the number of threads. The hollow markers indicate those of PSkyline, denoted by Px for using x threads. Note that, comparing when number of threads is four, both algorithms achieve up to 3-times speedup from those when number of threads is one. Though such speedup decreases, when comparing numbers of threads are four and eight, the overall performances monotonically improves as the number of threads increases and VSkyline consistently outperforms PSkyline, which indicate that our proposed method for dominance tests in VSkyline are improving performances, without hindering the inherent effectiveness of PSkyline.

5. CONCLUSIONS

In this paper, we proposed an efficient vectorization for skyline computation, exploiting data-level parallelism of dominance tests by using SIMD architectures. We first showed that a naive vectorization of dominance tests performs even worse than a non-vectorized version. The naive vectorization sequentially handled packed conditions for one dimension as in the non-vectorized version, incurring the increase of extra instructions to manipulate dominance tests. To address this problem, VSkyline deals with packed conditions as a whole, which is able to reduce the number of both branch instructions and branch mispredictions. VSkyline could thus boost the performance by up to three folds. In addition, VSkyline can be more efficient as the parallelism degree in future CPUs increases, and the idea of handling packed conditions can shed light on other similar applications.

ACKNOWLEDGMENTS

We are grateful to Dr. Sungwoo Park for providing PSkyline source code. This work was supported in part by MKE, Korea under ITRC NIPA-2010-(C1090-1021-0008)and MEST, Korea under NRF Grant (NRF-2009-0084870). This research was supported by National IT Industry Promotion Agency (NIPA) under the program of Software Engineering Technologies Development.

6. REFERENCES

[1] A. Ailamaki, D. J. DeWitt, M. D. Hill, and D. A. Wood. DBMSs on a modern processor: Where does time go? In *VLDB*, pages 266–277, 1999.

[2] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*, pages 421–430, 2001.

[3] J. Chomicki, P. Godfery, J. Gryz, and D. Liang. Skyline with presorting. In *ICDE*, pages 717–719, 2003.

[4] P. Godfrey, R. Shipley, and J. Gryz. Maximal vector computation in large data sets. In *VLDB*, pages 229–240, 2005.

[5] D. Kossmann, F. Ramsak, and S. Rost. Shooting stars in the sky: An online algorithm for skyline queries. In *VLDB*, pages 275–286, 2002.

[6] K. C. Lee, B. Zheng, H. Li, and W.-C. Lee. Approaching the skyline in Z order. In *VLDB*, pages 279–290, 2007.

[7] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progessive algorithm for skyline queries. In *SIGMOD*, pages 467–478, 2003.

[8] S. Park, T. Kim, J. Park, J. Kim, and H. Im. Parallel skyline computation on multicore architectures. In *ICDE*, pages 760–771, 2009.

[9] D. A. Patterson and J. L. Hennessy. *Computer Organization and Design: The Hardware/Software Interface(4th edition)*. Morgan Kaufmann Publishers Inc., 2008.

[10] J. Reinders. *VTune Performance Analyzer Essentials*. Intel Press, 2005.

[11] K. Tan, P. Eng, and B. C. Ooi. Efficient progressive skyline computation. In *VLDB*, pages 301–310, 2001.

[12] A. Vlachou, C. Doulkeridis, and Y. Kotidis. Angle-based space partitioning for efficient parallel skyline computation. In *SIGMOD*, pages 227–238, 2008.

[13] S. Zhang, N. Mamoulis, and D. W. Cheung. Scalable skyline computation using object-based space partitioning. In *SIGMOD*, pages 483–494, 2009.

[14] J. Zhou and K. A. Ross. Implementing database operations using SIMD instructions. In *SIGMOD*, pages 145–156, 2002.

On Models and Query Languages for Probabilistic Processes *

Daniel Deutch Tova Milo
Tel Aviv University
danielde@post.tau.ac.il milo@post.tau.ac.il

ABSTRACT

Probabilistic processes appear naturally in various contexts, with applications to Business Processes, XML data management and more. Many models for specifying and querying such processes exist in the literature; a main goal of research in this area is to design models that are expressive enough to capture real-life processes and analysis tasks, but at the same time allow for efficient query evaluation. We depict the model established in [13, 16, 17, 18], and claim that it achieves a good balance between expressivity and query evaluation complexity. We compare and contrast the model with other common models for probabilistic processes, highlighting the different choices made in models design and their effect on expressivity and incurred complexity.

1. INTRODUCTION

Probabilistic processes, i.e. processes with some probability distribution over their possible executions, appear naturally in various contexts. For instance, the possible behaviors of web-site users may be captured by such processes, where the probabilities are derived from the popularity of actions among previous users; a recent paper [6] explains how to capture probabilistic XML data via a probabilistic process specification; and there are many additional applications e.g. in areas such as Natural Language Processing [29] or computational biology [35]. Probabilistic processes occurring in such cases typically have intricate structures, and may induce a large (or even infinite, in presence of recursion) number of possible executions [6, 15].

The importance of probabilistic processes along with their complex nature call for the use of a *query language*, that will allow to analyze the processes and their possible executions. The results of such analysis are then applied for debugging the process, optimizing it, or identifying optimal ways to use it. Examples for analysis tasks include *termination* analysis (i.e. finding the probability that the process terminates), identifying likely execution flows (or parts thereof), identifying the probability that the execution reaches a given point, and so on.

To allow for an analysis of the above flavor, three components are required. First, one should design a formal *model* for probabilistic processes and their executions. Second, the model should be accompanied by a corresponding query language that will allow to specify analysis tasks over such processes and executions; the model and query language should be expressive enough to capture real-life processes and analysis tasks, and simple enough to allow for effective formulation of process specification and queries. Third, the model and query language must be supported by efficient query evaluation algorithms. Clearly, there exists a tradeoff between the expressive power of the model and query language, and the complexity of query evaluation. An important goal of research in this area is to find a good balance between the two.

Many models and query languages for probabilistic processes (e.g. [4, 6, 9, 17, 25, 30, 34]) appear in the literature, and vary in their expressive power and the query evaluation complexity they admit. Specifically, in [5, 16, 17] we have suggested a model and query language for *Probabilistic Business Processes (BPs)*, which are used to depict the logical structure of business activities that are common e.g. in e-commerce and in Web Applications [5, 19]. The model is an abstraction of the BPEL [8] standard and allows for an intuitive, graphical representation of the process. The query language is then based on the same graph-based view and allows users to query processes visually, in an intuitive manner, very analogous to how the processes are typically specified. We then studied query evaluation, and provided efficient algorithms, for different fragments of the query language [13, 16, 17].

We claim that our proposed model is successful in

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

achieving a good balance between the expressivity and the complexity of query evaluation. To substantiate this claim, we first informally depict the model and the complexity results achieved for query evaluation. Then, we review other common models and compare their properties. Via this comparison, we highlight the different choices that are made in the models design, and their effects on expressivity and query evaluation complexity.

We next provide a brief overview of the Business Process model and query language.

A Business Process (BP) specification is abstractly modeled as a nested DAG consisting of activities (nodes), and links (edges) between them, that detail the execution order of the activities [15, 23]. For example, consider a BP of an on-line travel agency. The BP specification may include activities (nodes) for flight and hotel reservation, car rental, payment and confirmation services, and edges that describe their execution order. The DAG shape allows to describe parallel computations. For instance, advertisements may be injected in parallel to the search. BP activities may be either atomic, or compound. In the latter case their possible internal structures, called *implementations*, are also detailed as DAGs, leading to the nested DAG structure. A compound activity may have different possible implementations, corresponding to different user choices, variable values, servers availability, etc. An *Execution Flow* (abbr. EX-flow) is then an actual running instance of a BP, obtained by choosing a single implementation for each compound activity. A BP specification induces a set of such possible EX-flows; this set may be large, or even infinite when the BP specification contains *recursion*.

In practice, some EX-flows are more common than others. This is modeled by a probability distribution over the possible implementations of compound activities [16, 17]. A BP specification along with a description of such distribution is called a *Probabilistic BP*. We note that the probabilities of choices dictating the execution course are, in typical cases, *inter-dependent*; for instance a user making a reservation for a flight to Paris is more likely to also reserve an hotel in Paris. We allow to model such dependencies in a straightforward manner.

We also defined a query language for analyzing BPs. Queries are used to define EX-flows or parts of them, that are of interest to the analyst. For our travel agency example, an analyst may wish to find out *how is one likely to book a travel package containing a flight reservation?*, or *how is this likely to be done for travelers of a particular airline company, say British Airways?*. There may be many different ways for users to book such travel packages. But assume, for instance, that we obtain that a likely scenario for British Airways reservations is one where users first search for a package containing both flights and hotels, but eventually do not reserve an hotel. Such a result may imply that the combined deals suggested for British Airways fliers are unappealing, (as users are specifically interested in such deals, but refuse those presented to them), and can be used to improve the Web site. Queries are defined using *execution patterns* (abbr. EX-patterns), generalizing EX-flows similarly to the way tree patterns, used in query languages for XML, generalize XML trees [10]. In more details, EX-patterns bear the structure of an EX-flow, where activity names are either specified, or left open using a special ANY symbol and then may match any node. Edges in a pattern are either regular, interpreted over edges, or transitive, interpreted over paths. Similarly, activities may be regular or *transitive*, for searching only in their direct internal flow or for searching in any nesting depth, respectively. A match of the query is captured via the notion of an *embedding*, which is a homomorphism from an EX-pattern to an EX-flow, respecting node labels and edge relation. We then note that given a BP specification s and a query q, the number of possible EX-flows of s that qualify according to q may be extensively large, or even infinite when s is recursive. In practice, analysts are only interested in the possible EX-flows of s that are most likely to occur in practice. This is in fact a flavor of *top-k analysis*.

In many cases, analysts are further interested only in some *part* of the possible EX-flows. For that, we define top-k *projection* queries, whose output consists of the likely sub-flows. An important question that rises when considering projection queries is the choice of a ranking metric for the query results. Recall that our model associates a likelihood value with every possible EX-flow of the BP specification; with projection queries, a single projection results may originate from multiple EX-flows. In this case, the likelihoods of all such EX-flows should then be *aggregated*, to form the result score. Evidently, different choices of such aggregation functions require different query evaluation mechanisms and incur different complexity of query evaluation. We consider the *max* and *sum* aggregation functions, explain their intuitive meaning and depict the complexity of query evaluation for each choice of function.

Paper Organization. In Section 2 we recall our model of probabilistic BPs and queries over such BPs; in Section 3 we overview the complexity of query evaluation for such queries. In Section 4 we overview other common models for probabilistic models, comparing them to our BP model. We conclude in Section 5.

2. MODEL

We (informally) depict in this section the model of probabilistic Business Processes (BPs), and a query language over such processes, originally defined and used in [5, 16, 17, 18]. Additional process models and their connection to this model are discussed in Section 4.

2.1 Business Processes

We start by depicting the model for Business Processes (without probabilities), then introduce probabilities. At a high-level, a BP specification encodes a set of activities and the order in which they may occur. A BP specification is modeled as a set of node-labeled DAGs. Each DAG has a unique start node with no incoming edges and a unique end node with no outgoing edges. Nodes are labeled by activity names and directed edges impose ordering constraints on the activities. Activities that are not linked via a directed path are assumed to occur in parallel. The DAGs are linked through implementation relationships; the idea is that an activity a in one DAG is realized via the activities in another DAG. We call such an activity *compound* to differentiate it from *atomic* activities which have no implementations. Compound activities may have multiple possible implementations, and the choice of implementation is controlled by a a condition referred to as a *guarding formula*.

EXAMPLE 2.1. *Fig. 1 depicts a partial BP specification. Its root DAG consists of a single activity* `chooseTravel`. `chooseTravel` *is a compound activity having 3 possible implementations* F_2, F_3, F_4. *These correspond to different choices of travel search (flights only, flights + hotels, or flights + hotels + cars) and are guarded by corresponding formulas. The idea is that exactly one formula is satisfied at run-time (the user chooses one of the three search types) and thus* `chooseTravel` *is implemented either by* F_2, F_3 *or* F_4. *Consider for example* F_1*; it describes a group of activities comprising user login, the injection of an advertisement, the* `Flights` *activity, and the* `Confirm` *activity. Directed edges specify the order of activities occurrence, e.g. users must login before choosing a flight. Some of the activities (e.g.* `Advertise` *and* `Flights`*) are not in a particular order, and thus may occur* in parallel. `Login` *and* `Advertise` *are atomic whereas* `Flights` *and* `Confirm` *are compound. Note that the specification is recursive as e.g.* F_2 *may call* F_1.

We note that satisfaction of guarding formulas is determined by external factors, such as user choices. We assume that exactly one guarding formula can be satisfied when determining the implementation of a given compound activity occurrence, but satisfaction of guarding formulas can change if activities occur several times. For instance, a user may choose to search for flights and hotels the first time she goes through F_1 and for flights only the second time.

Execution Flows. An Execution Flow (EX-flow) is modeled as a nested DAG that represents the execution of activities from a BP. Since, in real-life, activities are not instantaneous, we model each occurrence of an activity a by two a-labeled nodes, the first standing for the activity *activation* and the second for its *completion* point. These two nodes are connected by an edge. The edges in the DAG represent the ordering among activities activation/completion and the implementation relationships. To emphasize the nested nature of executions, the implementation of each compound activity appears in-between its activation and completion nodes. An EX-flow structure must adhere to the structure of the BP specification, i.e., activities must occur in the same order and implementation relationships must conform to τ.

EXAMPLE 2.2. *Two EX-flows of the travel agency BP are given in Fig. 2. Ordering edges (implementation edges) are drawn by regular (resp. dashed) arrows. Each EX-flow describes a sequence of activities that occurred during the BP execution. In Fig. 2(a) the user chooses a "flights+hotels" search, reserving a "British Airways" flight and a "Marriott" hotel, then confirms. Fig. 2 (b) depicts another possible EX-flow, where the user chooses a "flights only" search, followed by a choice of British Airways flight, but then resets and makes other choices (omitted from the figure).*

Likelihood. We note that the EX-flow occurring at run-time is dictated by the truth values of guarding formulas, which in turn dictate the choice of implementation for compound activities (out of the possibly multiple implementations associated with each such activity in the BP specification). In a probabilistic process, each such guarding formula is associated with a probability of being satisfied at run-time, and this probability may be dependant

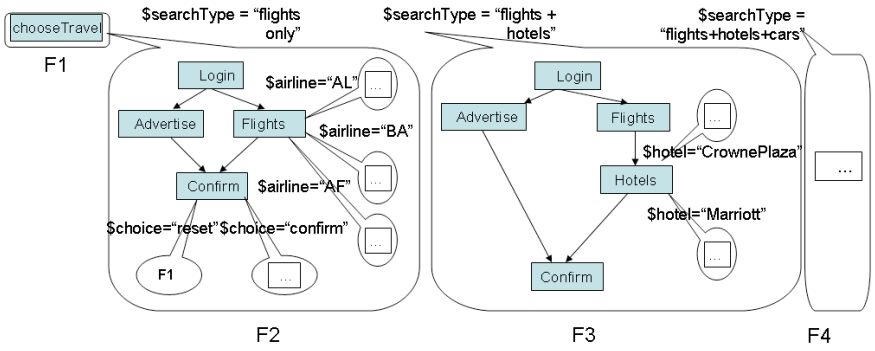


Figure 1: Business Process Specification

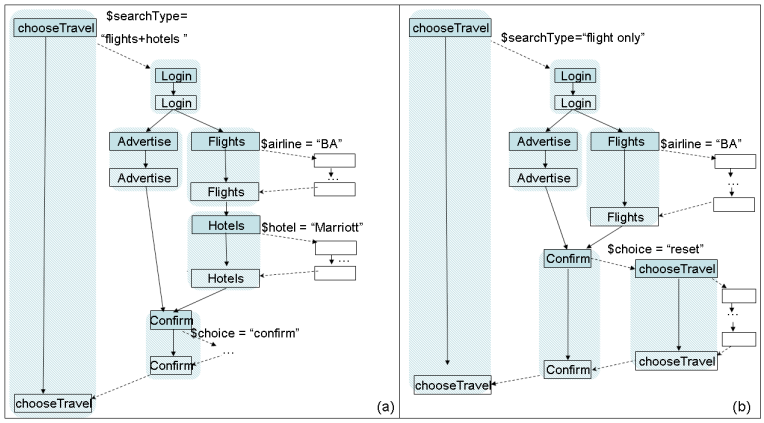


Figure 2: Execution Flows

on the EX-flow thus far and formulas previously satisfied.

To model this we use two likelihood functions. The first, named c–likelihood (choice likelihood), associates a probability value with each guarding formula (implementation choice), describing the probability that the formula holds, *given the partial EX-flow that preceded it.* The second, named f–likelihood (flow likelihood) reflects the joint likelihood of satisfaction of guarding formulas occurring along the flow and is defined as the multiplication of c–likelihood values. A BP specification along with a likelihood function over the guarding formulas occurring in the specification is called a *Probabilistic Business Process.*

Note that the c–likelihood function δ receives two inputs: a guarding formula f in hand, and the partial EX-flow e' that preceded the implementation choices guarded by f. This allows us to capture *dependencies* among likelihood values. We distinguish between three classes of c–likelihood functions, according to their sensitivity to e' (referred to as "history").

History independence: History-independent functions imply probabilistic independence between the choices of implementation in different activities occurrences throughout the flow. More formally, a c–likelihood function is history-independent for every guarding formula f, c–likelihood (e,f) = c–likelihood (e',f) for each two EX-flows e, e'.

Bounded-history: Bounded-history functions capture the more common scenario where the c–likelihood value does depend on the history e of the EX-flow, but only in a bounded manner. That is, to determine the c–likelihood of any implementation choice at an activity node n, it suffices to consider only the implementation choices of the last b preceding compound activities, for some bound b. By "preceding" we refer here to activity nodes $\hat{n}$ that are *ancestors* of n in e (in contrast to nodes that occur in parallel and thus in general may or may not precede n). By "choice" we refer to the formula guarding the implementation selected for the activity.

Unbounded-history: Unbounded-history functions may use an *unbounded* portion of the flow history

e to compute the next choice's likelihood. For instance, if the price of a given product depends on the *exact full sequence of searches that the user performed prior to the purchase*, then the corresponding c–likelihood function is unbounded-history.

Observation. Note that, by definition, for non-recursive BPs, c–likelihood functions are always bounded-history, with the bound being, at most, the BP nesting depth. Recursive BPs, on the other hand, may have unbounded-history c–likelihood functions. In practice, typical c–likelihood functions are bounded-history, and moreover the bound is typically small [36].

2.2 Query Language

We next consider a query language for Probabilistic Business Processes, originally defined in [5] and further refined in [17, 18]. We start by considering *selection queries*, whose output is the (representation of the) set of all EX-flows of the original BP specification, that also match the query. We then introduce *projection queries*, that further allow to focus on some sub-flows that are of interest.

Queries are defined using *execution patterns*, an adaptation of the tree-patterns of existing XML query languages, to nested EX-flow DAGs. Such patterns are similar in structure to EX-flows, but contain *transitive edges* that match any EX-flow *path*, and *transitive activity nodes*, for searching deep within the implementation subgraph, of the corresponding compound activities, at any level of nesting. Nodes/ formulas in the pattern may be labeled by the wildcard ANY and then may match any EX-flow node/formula.

EXAMPLE 2.3. *The query in Fig. 3 (a) (ignore for now the rectangle surrounding a sub-graph of the pattern) describes EX-flows that contain a choice of some British Airways (abbr. "BA") flight followed by a confirmation. The double-lined edges are* transitive edges*, and may match any sequence of activities. The doubly bounded* `chooseTravel` *activity is a* transitive activity*. Its implementation nodes may be embedded anywhere inside the implementation of the corresponding EX-flow activity, at any nesting level.*

The matching of an EX-pattern p to an EX-flow e is called an *embedding*. An embedding of p into e is a homomorphism ψ from nodes and edges in p to nodes,edges and paths in e such that the root of p is mapped to the root of e; activity pairs in p are mapped to activity pairs in e, preserving node labels and formulas; a node labeled by ANY may be mapped to nodes with any activity name. For non-transitive compound activity pairs in p, nodes in their direct implementation are mapped to nodes in the direct implementation of the corresponding activity pair in e. As for edges, each (transitive) edge from node m to n in p is mapped to an edge (path) from $\psi(m)$ to $\psi(n)$ in e.

We then distinguish between selection and projection queries, as follows.

Selection Queries. A selection query is represented by an EX-pattern. An EX-flow e belongs to the query result if there exists some embedding of p into e. We then say that e *satisfies* p. When evaluated against a Business Process s, the output of p consists of all full EX-flows of s that also satisfy p; the set of top-k most likely out of these EX-flows are denoted top-k(s,p).

Projection Queries. In many cases, analysts are not interested in full execution flows of the process, but rather in focusing on some part of them. To that end, we define projection queries over probabilistic processes. For selection queries, the query result consisted of the full EX-flow in which the EX-pattern was embedded. We generalize the definition of such queries and allow a specific part of the pattern to be specified as the *projected part*. The rest of the pattern serves for selecting EX-flows of interest. Namely, only EX-flows in which an embedding of the entire pattern are considered; within these flows, only nodes and edges matching the projected part are in fact projected out and appear in the query result.

Given an embedding ψ of a query q in an EX-flow e, the embedding result is then defined as the nodes and edges of e to which ψ maps the nodes and edges of the projected part of q. For an EX-flow e, the result of q on e, denoted $q_{\downarrow}(e)$, consists of the results of all possible embeddings of q into e; finally, for a BP s, the result of q on s, denoted $q_{\downarrow}(s)$ is the union of all possible results of q when applied on the EX-flows of s. Namely $q_{\downarrow}(s) = \bigcup_{e \in flows(s)} q_{\downarrow}(e)$ where flows(s) is the set of possible EX-flows of s.

Note that an EX-flow $e' \in q_{\downarrow}(s)$ may originate from several EX-flows of s, namely there may be several $e \in flows(s)$ s.t. $e' \in q_{\downarrow}(e)$. Before defining the top-k projection results, we should first decide on how to aggregate the weights of these individual EX-flows, to form the score of projection result. We consider two possible aggregation functions (and consequently, semantics of queries), *max* and *sum*. Under *max* semantics, the *score* of e' is defined as $score(e') = max\{f\text{-}likelihood(e) \mid e \in flows(s) \wedge e' \in q_{\downarrow}(e)\}$. Under *sum* semantics,

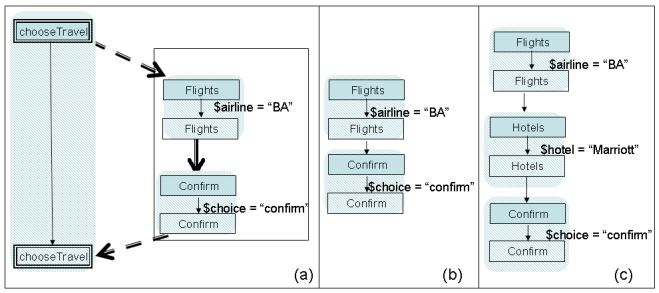


Figure 3: Query

$score(e') = \sum_{e \in flows(s) \wedge e' \in q_{\downarrow}(e)} flikelihood(e)$.

The top-k results of a projection query q over a BP specification s, with respect to *max* (*sum*) semantics, are then denoted top-$k_{max}(q_{\downarrow}, s)$ (top-$k_{sum}(q_{\downarrow}, s)$). When the semantics used is clear from context, we omit it from notation and simply write top-$k(q_{\downarrow}, s)$.

EXAMPLE 2.4. *Let us consider again the EX-pattern in Figure 3(a), this time as a query, with the rectangle denoting its projected part. Note that the projection focuses on the execution sub-flows that may occur between the point where a user chooses a "BritishAirways" flight and the final confirmation of her reservation. Note that, due to recursive nature of the BP, there are in general an infinite number of such possible sub-flows (query answers) - a user may reset and restart the search an unbounded number of times. Two possible answers to the query appear in Fig. 3 (b) and (c). The first answer corresponds to users that choose at some point a "flightsOnly" search, pick a "BA" flight and then immediately confirm. The second answer corresponds to users that choose at some point a "flights+hotels" search, pick "BA" as airline and Marriott as hotel, and confirm.*

Observe that each of these answers (sub-flow) have potentially an infinite number of possible origin EX-flows, differing in what the user had done prior to/after the reservation. The likelihood of each answer is the max / sum of likelihoods of all these origins, according to the semantics in hand.

To illustrate the difference between the two semantics (max and sum), consider the above query and a case where one particular deal *D, consisting of a specific flight, hotel and car rental reservation, is very popular, but where packages consisting of only flight and hotel reservations (with no car rental) are* overall *more common (even though each given offer is individually less popular than the specific flight+hotel+car deal D). With* sum *semantics, the result corresponding to the flight+hotel option is ranked highest, as it appears in most EX-flows. But with* max *semantics, flight+hotel+car will be ranked highest, as there exists one very popular EX-flow in which it appears.*

3. QUERY EVALUATION

We next review the main results on query evaluation over probabilistic Business Processes. We omit the proofs and algorithms and refer the reader to prior publications for the full details.

There are three axes that determine the complexity of query evaluation: whether the BP specification is recursive or not, the class of the c–likelihood function used in the process description (i.e. its level of dependency over the history), and the query semantics (selection vs. projection). Interestingly, when considering projection queries, the choice of aggregation function also bears a significant effect on the complexity of query evaluation. The complexity results are summarized in Table 1.

We start the discussion with selection queries, then proceed to projection queries.

3.1 Selection Queries

Given a BP specification s, a selection query q, and a number k, the top-k results of q with respect to s (denoted $top{-}k(q, s)$) are defined as the k most likely flows of s, out of those in which an embedding of q exists. [1]. We refer to the problem of identifying top-k(q,s) (given the above input) as `TOP-K-ANSWERS`.

We next discuss the problem complexity for the different classes of c–likelihood functions.

History-independent c–likelihood function. When the c–likelihood function is history-independent, we can design an efficient algorithm for top-k selection query evaluation, as the following theorem

[1]Certain EX-flows may have equal weights, which implies that there may be several valid solutions to the problem, in which case we pick one arbitrarily.

BP	Weight Function	Queries	Data Complexity	Query Complexity
(Non-)Recursive	History-independent	Selection / Projection (max)	PTIME [16, 17, 18]	EXPTIME, NP-hard [14]
(Non-)Recursive	Bounded-History	Selection / Projection (max)	PTIME in BP, EXPTIME (NP-hard) in bound [16, 17]	EXPTIME
Recursive	Unbounded-History	Selection / Projection	Undecidable [16, 17]	Undecidable
(Non-)Recursive	History-independent	Projection (sum)	EXPTIME, NP-hard [13]	EXPTIME
Non-Recursive	History-independent	Projection (sum) restricted	PTIME (unit-cost rat. arithmetic) [13]	EXPTIME
(Non-)Recursive	Bounded-History	Projection (sum)	EXPTIME in BP, 2-EXPTIME in bound [13]	EXPTIME

Table 1: Complexity of query evaluation for Business Processes

holds (the algorithm proving the theorem correctness originally appeared in [16] and was improved in [18]).

THEOREM 3.1. *[16, 18] Given a probabilistic BP s with a history-independent c–likelihood function, and a query q, we may solve* `TOP-K-ANSWERS` *in time polynomial in the size of s and in the output size, and exponential in the query size.*

The algorithm solving `TOP-K-ANSWERS` works by "intersecting" the BP specification s with the query q, outputting a BP specification s' whose EX-flows are exactly the EX-flows of s in which there exists an embedding of q. The algorithm then employs a sophisticated A*-style analysis that explores the space of possible EX-flows of s', by repeatedly testing possible expansions of activities and avoiding infinite loops by maintaining a table of the previously computed sub-flows in a dynamic programming style.

We can also show that a PTIME algorithm w.r.t. query size is not possible, unless P=NP. For that, we define the corresponding decision problem `BEST-ANSWER`, which tests, whether the top-1 EX-flow of a given BP specification is more likely than some threshold t. The following theorem holds (the proof appears in [17], following a construction from [14], using reduction from 3-SAT):

THEOREM 3.2. *[14, 17]* `BEST-ANSWER` *is NP-hard when the input size is considered to be the query size (and the BP specification size is considered a constant).*

Bounded-History c–likelihood function. Bounded-History c–likelihood functions pose further challenges, as the c–likelihood of each choice may depend on a number of other choices. The following theorem was shown in [17]:

THEOREM 3.3. *[17] Given a BP s, a bounded-history c–likelihood function δ with bound b, and a query q, we may solve* `TOP-K-ANSWERS` *in time polynomial in the size of s and in the output size, and exponential in b and in the query size.*

The general idea of the algorithm is to create, a new BP s', with a new, *history-independent*, c–likelihood function δ', such that s and s' have essentially the same set of flows with the same f–likelihood . Then, we apply the algorithm from the proof of Theorem 3.1. To obtain s' and δ' we annotate the activity names in the specification, "factoring" within the names all information required for the computation of c–likelihood of formulas, namely a pre-condition vector, including the m last choices for all activities. Additionally, the new activities names also contain post-condition vectors, necessary to assure consistencies between pre-conditions assumed by activities and what has happened in their predecessors.

It was further shown in [17] that the added exponential dependency on the history-bound b is inevitable, as the following theorem holds (proof by reduction from Set Cover):

THEOREM 3.4. *[17] For bounded-history c–likelihood functions with bound b,* `BEST-ANSWER` *is NP-hard when the number of activities in the BP specification and in the query are considred to be constants, and b is considered to be the input size.*

Unbounded-History c–likelihood functions. Last, for unbounded-history c–likelihood functions, we showed in [16] that `TOP-K-ANSWERS` becomes impossible to solve, as the following theorem holds:

THEOREM 3.5. *[16] If the c–likelihood function given as input may be unbounded-history, then* `BEST-ANSWER` *is undecidable.*

The proof is by reduction from the halting problem, where we encode a Turing Machine as a BP specification with unbounded-history c–likelihood function; the latter is used to model the TM tape.

3.2 Projection Queries

We next turn to projection queries. It is easy to observe that all hardness results presented above for selection queries, also hold for projection queries. However, it turns out that the upper bounds depend upon the aggregation function in use. Specifically, for the *max* semantics:

THEOREM 3.6. *[17] Theorems 3.1 and 3.3 hold for projection queries with* max *semantics.*

To prove Theorem 3.6 we adapt the algorithms that were designed for selection queries to account for projection queries with max semantics. A naive attempt for such adaptation involves selection followed by projection, namely the generation of a BP specification whose EX-flows are the top-k selected EX-flows, then projecting these EX-flows to find the top-k projections. However the size of the selected EX-flows, and consequently the algorithm complexity is exponential in the size of the input BP specification, even when the projection results themselves are of small size. Moreover, we may show that it is infeasible to compute a BP specification whose EX-flows correspond to *all* projection results and then retrieve the top-k results out of them, intuitively because projection over transitive edges may result in exponentially many paths which cannot be compactly represented together. However, we may still perform a two-steps algorithm, similar to the one employed for selection queries: he first step of the refined evaluation algorithm generates a specification that captures only a *subset* of the projection results, where for each transitive edges only the top-k matching paths are kept; this subset includes in particular the top-k projections. We can then find the top-k EX-flows of this specification, which are the top-k projection results with respect to max semantics.

When the *sum* aggregation function is used, query evaluation becomes much more difficult. Specifically, we can show the following:

THEOREM 3.7. *[13] For projection queries with* sum *semantics,* `BEST-ANSWER` *is NP-hard (under Turing computation model) in the BP size, even for non-recursive specifications and for queries with no transitive nodes.*

The proof is by reduction from 3-SAT. Interestingly, the problem is hard even for non-recursive BPs. This is because the hardness is due to the DAG structure of the BP graphs, that may lead to exponentially many paths being projected over the query transitive edges; consequently the sum of likelihoods of exponentially many EX-flows must be computed. See the comparison in Section 5 to Probabilistic XML.

On the other hand, under certain plausible assumptions over the BP specification structure, we can design an algorithm whose complexity is EXPTIME in the BP specification size. We say that a projection query q w.r.t. a BP specification s has *separated likelihoods*, if for every two query answers, either they have equal likelihoods or the difference of their likelihoods is greater than some fixed constant ϵ. The following theorem then holds:

THEOREM 3.8. *[13] For projection queries with sum semantics and BP specifications with history-independent c–likelihood function:,* `TOP-K-ANSWERS` *may be solved in EXPTIME, for: (1) non-recursive BP specifications, and (2) recursive BP specifications, when the query has separated likelihoods w.r.t. the specification.*

The algorithm uses a small world theorem, showing that among the infinitely many paths that may match the query transitive edges, it is sufficient to examine paths whose length is bounded by the size of the BP specification multiplied by k (the number of required results); consequently only exponentially (in the above quantity) many paths are examined. Then, the algorithm reduces `TOP-K-ANSWERS` to the computation of likelihoods for a set of exponentially many candidate answers that are represented as boolean queries, i.e. queries for which we ask only for the likelihood of a match; finally, techniques from [25] are adapted to approximate the likelihood values of these boolean queries up to a point allowing to separate the different candidate answers (this is why separated likelihoods are required). Exact computation of likelihood values for boolean queries is impossible here since they may in general be irrational [13], thus we must resort to their approximation.

Note that the proof for NP-hardness (Theorem 3.7) used queries with transitive edges. When no transitive edges are allowed in the query, our algorithm is more efficient, for non-recursive BP specifications. Specifically, consider the unit-cost rational arithmetic model [7]: in this model, when computing the complexity, we assume that each arithmetic operation on rational numbers may be done in O(1), regardless of the numbers size. The following theorem then holds:

THEOREM 3.9. *[13] When the query projected part does not include transitive edges, and the BP specification is non-recursive,* `TOP-K-ANSWERS` *for projection queries with* sum *semantics may be solved*

in PTIME under the unit-cost rational arithmetic model.

The assumption of unit-cost rational arithmetic is necessary here, as there are examples where even non-recursive BP specifications and queries without transitive edges yield projection results with probabilities whose encoding is exponential in the BP specification size.

When the c–likelihood is bounded-history, we may first use the construction depicted above for proving Theorem 3.3 to obtain a new BP specification s' with history-independent c–likelihood function, then apply the above algorithm. Note that since the size of s' may be exponential in the history-bound, and the algorithm that finds the top-k projections may incur exponential time in the size of its input, the overall complexity is double-exponential in the size of the bound. The following theorem holds:

THEOREM 3.10. *[13] For projection queries with sum semantics and BP specifications with bounded-history c–likelihood function with bound b,* `TOP-K-ANSWERS` *may be solved in time exponential in the size of the BP specification and double exponential in b, for: (1) non-recursive BP specifications, and (2) recursive BP specifications, when the query has separated likelihoods w.r.t. the specification.*

4. ADDITIONAL MODELS

We have depicted above our model for probabilistic Business Processes. The literature contains a variety of formalisms for *probabilistic process specifications*. We next discuss a representative subset of these models; for each model we review the query languages used for its analysis, and the corresponding complexity results for query evaluation. A summary of the models and results reviewed in this Section appears in Table 2. We also compare and contrast these models with the model of probabilistic BP.

4.1 Markov Chains

The probabilistic counterpart of Finite State Machines (FSMs) is called Markov Chains (MCs) [30]. MCs are in fact FSMs, but with transitions associated with probabilities; these state machines bear the Markovian property [30], meaning that the choice of transition is independent of the previous states that were traversed, given the current state. The common approach in analysis of such processes is to use a query language based on *temporal logic* [33], e.g. LTL, CTL^* or μ-calculus (these differ from each other in terms of expressive power, see e.g. [33]). Query evaluation for such logics then correspond to computing the probability that the property defined by the given query, holds for a random walk over the Markov Chain. It is known that query evaluation of such temporal logic queries over a Markov Chain may be performed in time polynomial in the size of the Markov Chain, and exponential in the query size [11].

Comparison to the BP model and query language. Finite State Machines (and consequently, Markov Chains) lack the expressive power to capture the possibly recursive processes allowed in the BP model here (we shell see in the sequel that BPs are equivalent to "stronger" models). The definition of history-unbounded c–likelihood functions resembles (and is inspired by) the Markovian Property.

As for the query language, a first observation is that temporal logic allows only for *boolean queries* (i.e. computing the probability that a given boolean property holds), while selection and projection queries of the kind depicted in Section 2 were not studied in the context of Markov Chains (to the best of our knowledge). But even when considering just boolean queries, there are queries expressible in our language but not in temporal logic. To that end, note that temporal logics are *bisimulation-invariant*, intuitively meaning that they can query only the process "behavior" rather than its structure (see [14] for exact notions). In contrast, our query language takes a database approach and allows to further query the structure of the process as well as the structure of its possible executions. Indeed, we have shown in [14] that some queries expressible in our query language are not expressible in temporal logic (and vice versa).

4.2 Introducing Function Calls

As stated above, Markov Chains are the probabilistic counterpart of Finite State Machines. Similarly, there exist probabilistic counterparts to stronger model; these models allow for function calls, and in their full-fledged version, also for recursion. Specifically, *Recursive Markov Chains* (RMCs) [25] extend Markov Chains to allow for recursive invocations of procedures. An RMC consists of a collection of finite state components, each of which is a Markov chain, that can call each other in a potentially recursive manner. The authors of [25] further show that RMCs generalize (and can express) previous important models for probabilistic processes such as probabilistic Pushdown Automata [9], and Stochastic Context Free Grammars [32].

Model	Query	Data Complexity	Query Complexity
MC	Temporal	PTIME	EXPTIME
RMC	MSO	EXPTIME, at least as hard as SQRT-SUM	Non-elementary
HMC	MSO	PTIME (with unit-cost relational arithmetic)	Non-elementary
Tree-like HMC	MSO	PTIME	Non-elementary
Prob. XML [31, 37] (non-recursive)	Selection / Projection	PTIME	$\sharp P$-complete

Table 2: Complexity of query evaluation for different models

Query evaluation over RMCs was first studied in [25] for approximating the probabilities of reachability and termination. In a recent work, [6] has extended the analysis for a very strong query language, namely the Monadic Second Order (MSO) Logic. Denote as MSO-EVAL the problem of computing the probability that a given MSO query is satisfied in a random execution of an RMC; the following theorem then holds:

THEOREM 4.1. *[24] MSO-EVAL can be performed in EXPTIME Data Complexity and non-elementary query complexity.*

It is highly unlikely that query evaluation for RMCs may be solved in PTIME, as [25] showed that this problem is at least as hard as `SQRT-SUM`. `SQRT-SUM` is the problem of deciding, given natural numbers $(d_1, ..., d_n)$ and a natural number k, whether $\sum_{i=1,...,n} \sqrt{d_i} \leq k$, strongly believed not be solvable in PTIME under Turing Computation Model [27]. Interestingly, this hardness result holds even for very simple reachability and termination analysis, and even when we are only interested in approximating the probabilities.

Also note in this context the non-elementary query complexity of the evaluation problem. This complexity is entailed by the use of the highly expressive MSO logic as a query language, and is unavoidable even for weaker process models due to [38].

Several restricted variants of RMCs were studied in [6, 25]; specifically, HMCs (Hierarchical Markov Chains) is a restricted version that does not allow for recursive calls. Interestingly, query evaluation here is still harder than for non-hierarchical MCs. This is due to the fact that the probability of a given property may be so small that exponentially many bits are required to represent it. This, in turn is due to the function-like structure of HMCs that allow several functions to call a single function; in this way, exponentially many call paths may be represented compactly.

Thus, for the Turing computational model there is no hope for a PTIME data complexity algorithm. However, if we use the unit-cost rational arithmetic model [7], we do not have to care about the size of numbers. The following theorem then holds:

THEOREM 4.2. *[6] MSO-EVAL for HMC can be performed in PTIME Data Complexity (and non-elementary query complexity), under the unit-cost rational arithmetic model.*

If the HMC possible calls graph has a tree-shape, then the above problem is avoided and we obtain a PTIME data complexity under the conventional Turing Computational Model

THEOREM 4.3. *[6] MSO-EVAL for Tree-Like HMCs ([6]) can be performed in PTIME Data Complexity (and non-elementary query complexity), under the Turing Computational Model*

Comparison to the BP model and query language. Our Business Process model with history-independent likelihood functions may be shown to be equivalent to a restricted version of RMCs, namely 1-exit RMCs (the proof is an adaptation of Theorem 3 in [14] to a probabilistic context), and similarly non-recursive Business Processes are equivalent to 1-exit HMCs. To our knowledge, no extended model for RMCs that allows for dependencies in-between probabilistic choices was studied. As for the query language, our language (when queries are considered as boolean) is contained in MSO in terms of expressive power. But note that the use of MSO is very costly: it incurs non-elementary query complexity, with respect to the EXPTIME obtained for our language. Our query language is restricted, but is expressive enough to express interesting and practical properties as shown in [5], while allowing for a practically efficient evaluation. Furthermore, as the case for MCs, selection and projection queries were not studied in the context of RMCs (to our knowledge).

4.3 Probabilistic XML

It turns out that there are intricate connections between Probabilistic XML and probabilistic processes. Specifically, the recent work of [6] mentioned

above establishes a model for Probabilistic XML that is based on Recursive State Machines. Then, in addition to their study of MSO queries, they study more restricted languages which are common for XML querying such as (boolean) Tree Pattern Queries and XPath, and show that they incur lower query complexity: $FP^{\sharp P}$ (the class of computation problems that can be solved in polynomial time using a $\sharp P$ oracle) for tree pattern queries and PSPACE for XPath.

Previous works on Probabilistic XML documents [2, 3, 31, 37] used non-recursive models for expressing distributions over a finite collection of possible documents. In [31], a p-document is an XML tree with two types of nodes: ordinary nodes which are just regular XML nodes, and distributional nodes that define some probabilistic distribution over subsets of their children. A random XML document may then be generated by making probabilistic choices that follow these distributions. The authors of [31] then study query evaluation over p-document and use a language that allows for selection and projection; the semantics used there for projection is the counterpart of our *sum* semantics, and they show that query evaluation is feasible, as follows:

THEOREM 4.4. *[31] Query Evaluation over p-documents may be done in time complexity that is polynomial in the p-document size, and exponential in the query size.*

They also show that the exponential dependency on the query size is inevitable, unless $P = \sharp P$.

Comparison to the BP model and query language. An important distinction between the model of [31] and our BP model is that BP graphs are DAG-shaped while XML graphs are tree-shaped. Evidently, this difference causes an unavoidable (unless P=NP) exponential overhead in query evaluation, and also complicates the algorithmic construction (see [13] and the discussion in Section 3). While [6] considers XML with sharing, a model that does allow for DAG-shape graphs in the specification, the weaker query languages (tree patterns and XPath) studied in this cannot capture the DAG-shaped patterns expressible in our query language; also, [6] only studies boolean queries in contrast to the selection and projection queries allowed in our language.

5. CONCLUSION

We have reviewed in this paper several important models for specifying and querying probabilistic processes and compared them in terms of expressive power and complexity of query evaluation. We further depicted our proposed model for Business Process and our query language that allows for top-k selection and projection queries. We then explained how our proposed model achieves a reasonable balance between expressivity and evaluation complexity.

Clearly, more work has to be done to capture real-life probabilistic processes and analysis tasks. In particular, while the models consider, to some extent, the data manipulated in the process execution, more work is required for modeling and querying such manipulated data in real-life settings. The modeling (and analysis) of the data manipulated by processes was studied in e.g. [12, 20, 21, 22, 26, 28], but the processes studied there were not probabilistic; combining these two lines of research into a unified framework for probabilistic processes along with the data they manipulate. Additionally, we have discussed selection and projection queries, the latter with two particular aggregation functions (sum and max); in [1] the authors study aggregate queries for "monoid" aggregate functions (including sum, count, min, max) for Probabilistic XML. Studying further operations such as value-based joins and projection with different aggregation functions, in the context of the different models presented here is an intriguing challenge.

Acknowledgments

This research was partially supported by the Israeli Science Foundation (ISF), the US-Israel Binational Science Foundation (BSF), the EU Project Mancoosi and the Israel Ministry of Science Eshkol grant.

6. REFERENCES

[1] S. Abiteboul, T.H. Hubert Chan, E. Kharlamov, W. Nutt, and P. Senellart. Aggregate queries for discrete and continuous probabilistic XML. In *ICDT*, 2010.

[2] S. Abiteboul, B. Kimelfeld, Y. Sagiv, and P. Senellart. On the expressiveness of probabilistic XML models. *VLDB J.*, 18(5), 2009.

[3] S. Abiteboul and P. Senellart. Querying and updating probabilistic information in XML. In *Proc. of EDBT*, 2006.

[4] R. Alur, M. Benedikt, K. Etessami, P. Godefroid, T. Reps, and M. Yannakakis. Analysis of recursive state machines. *ACM Trans. Program. Lang. Syst.*, 27(4), 2005.

[5] C. Beeri, A. Eyal, S. Kamenkovich, and T. Milo. Querying business processes. In *Proc. of VLDB*, 2006.

[6] M. Benedikt, E. Kharlamov, D. Olteanu, and P. Senellart. Probabilistic XML via Markov chains. *PVLDB*, 3(1), 2010.
[7] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and real computation*. Springer-Verlag, 1998.
[8] Business Process Execution Language for Web Services. http://www.ibm.com/developerworks/library/ws-bpel/.
[9] T. Brazdil, A. Kucera, and O. Strazovsky. On the decidability of temporal properties of probabilistic pushdown automata. In *Proc. of STACS*, 2005.
[10] D. Chamberlin. Xquery: a query language for XML. In *Proc. of SIGMOD*, 2003.
[11] C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *J. ACM*, 42(4), 1995.
[12] T. Deng, W. Fan, L. Libkin, and Y. Wu. On the aggregation problem for synthesized web services. In *Proc. of ICDT*, 2010.
[13] D. Deutch. *Querying Web Applications Under Models of Uncertainty*. PhD thesis, Tel Aviv University, 2010. http://www.cs.tau.ac.il/ danielde/PhdThesis.pdf.
[14] D. Deutch and T. Milo. Querying structural and behavioral properties of business processes. In *Proc. of DBPL*, 2007.
[15] D. Deutch and T. Milo. Type inference and type checking for queries on execution traces. In *Proc. of VLDB*, 2008.
[16] D. Deutch and T. Milo. Evaluating top-k queries over business processes. In *Proc. of ICDE*, 2009.
[17] D. Deutch and T. Milo. Top-k projection queries for probabilistic business processes. In *Proc. of ICDT*, 2009.
[18] D. Deutch, T. Milo, N. Polyzotis, and T. Yam. Optimal top-k query evaluation for weighted business processes. In *Proc. of VLDB*, 2010.
[19] D. Deutch, T. Milo, and T. Yam. Goal-oriented web-site navigation for on-line shoppers. In *Proc. of VLDB*, 2009.
[20] A. Deutsch, M. Marcus, L. Sui, V. Vianu, and D. Zhou. A verifier for interactive, data-driven web applications. In *Proc. of SIGMOD*, 2005.
[21] A. Deutsch, L. Sui, V. Vianu, and D. Zhou. Verification of communicating data-driven web services. In *Proc. of PODS*, 2006.
[22] A. Deutsch and V. Vianu. Wave: Automatic verification of data-driven web services. *Data Eng. Bull.*, 31(3), 2008.
[23] P. Diniz. Increasing the accuracy of shape and safety analysis of pointer-based codes. In *LCPC*, 2003.
[24] K. Etessami and M. Yannakakis. Algorithmic verification of recursive probabilistic state machines. In *Proc. of TACAS*, 2005.
[25] K. Etessami and M. Yannakakis. Recursive Markov Chains, stochastic grammars, and monotone systems of nonlinear equations. *JACM*, 56(1), 2009.
[26] C. Fritz, R. Hull, and J. Su. Automatic construction of simple artifact-based business processes. In *Proc. of ICDT*, 2009.
[27] M. R. Garey, R. L. Graham, and D. S. Johnson. Some np-complete geometric problems. In *Proc. of STOC*, 1976.
[28] R. Hull and J. Su. Tools for composite web services: a short overview. *SIGMOD Rec.*, 34(2), 2005.
[29] D. Jurafsky, C. Wooters, J. Segal, A. Stolcke, E. Fosler G. Tajchman, and N. Morgan. Using a stochastic context-free grammar as a language model for speech recognition. In *Proc. of ICASSP*, 1995.
[30] J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. Springer, 1976.
[31] B. Kimelfeld and Y. Sagiv. Matching twigs in probabilistic XML. In *Proc. of VLDB*, 2007.
[32] K. Lary and S. J. Young. The estimation of stochastic context-free grammars using the inside-outside algrithm. *Computer, Speech and Language*, 4:35–56, 1990.
[33] Z. Manna and A. Pnueli. *The temporal logic of reactive and concurrent systems*. Springer-Verlag, 1992.
[34] T. Oates, S. Doshi, and F. Huang. Estimating maximum likelihood parameters for stochastic context-free graph grammars. In *Proc. of ILP*, 2003.
[35] T. Pavlidis. Linear and context-free graph grammars. *J. ACM*, 19(1), 1972.
[36] P. L. T. Pirolli and J. E. Pitkow. Distributions of surfers' paths through the world wide web: Empirical characterizations. *World Wide Web*, 2(1-2), 1999.
[37] P. Senellart and S. Abiteboul. On the complexity of managing probabilistic XML data. In *Proc. of PODS*, 2007.
[38] L. Stockmeyer. *The Complexity of Decision Problems in Automata Theory and Logic*. PhD thesis, Massachusetts Institute of Technology, 1974.

Business Intelligence for Small and Middle-Sized Entreprises

Oksana Grabova*
University of Lyon (ERIC Lyon2)
5 av P. Mendes-France
69676 Bron Cedex, France
oksana.grabova@eric.univ-lyon2.fr

Jerome Darmont
University of Lyon (ERIC Lyon2)
5 av P. Mendes-France
69676 Bron Cedex, France
jerome.darmont@univ-lyon2.fr

Jean-Hugues Chauchat
University of Lyon (ERIC Lyon2)
5 av P. Mendes-France
69676 Bron Cedex, France
jean-hugues.chauchat@univ-lyon2.fr

Iryna Zolotaryova
Kharkiv National University of Economics
9-a, pr. Lenina
61001 Kharkov, Ukraine
is@knue.edu.ua

ABSTRACT

Data warehouses are the core of decision support systems, which nowadays are used by all kind of enterprises in the entire world. Although many studies have been conducted on the need of decision support systems (DSSs) for small businesses, most of them adopt existing solutions and approaches, which are appropriate for large-scaled enterprises, but are inadequate for small and middle-sized enterprises.

Small enterprises require cheap, lightweight architectures and tools (hardware and software) providing online data analysis. In order to ensure these features, we review web-based business intelligence approaches. For real-time analysis, the traditional OLAP architecture is cumbersome and storage-costly; therefore, we also review in-memory processing.

Consequently, this paper discusses the existing approaches and tools working in main memory and/or with web interfaces (including freeware tools), relevant for small and middle-sized enterprises in decision making.

1. INTRODUCTION

During the last decade, data warehouses (DWs) have become an essential component of modern decision support systems in most companies of the world. In order to be competitive, even small and middle-sized enterprises (SMEs) now collect large volumes of information and are interested in business intelligence (BI) systems [26]. SMEs are regarded as significantly important on a local, national or even global basis and they play an important part in the any national economy [34]. In spite of multiples advantages, existing DSSs frequently remain inaccessible or insufficient for SMEs because of the following factors:

- high price;
- high requirements for a hardware infrastructure;
- complexity for most users;
- irrelevant functionality;
- low flexibility to deal with a fast changing dynamic business environment [56];
- low attention to difference in data access necessity in SMEs and large-scaled enterprises.

In addition, many projects fail due to the complexity of the development process. Moreover, as the work philosophies of small and large-scaled enterprises are considerably different, it is not advisable to use tools destined to large-scaled enterprises. In short, "one size does not fit all" [51]. Furthermore, there are a lot of problems in the identification of information needs of potential users in the process of building a data warehouse [7].

Thereby, SMEs require lightweight, cheap, flexible,

*This author also works at the Kharkiv National University of Economics, 9-a, pr.Lenina, 61001 Kharkov, Ukraine

simple and efficient solutions. To aim at these features, we can take advantage of light clients with web interfaces. For instance, web technologies are utilized for data warehousing by large corporations, but there is an even greater demand of such kind of systems among small and middle-sized enterprises. Usage of web technologies provides cheap software, because it eliminates the necessity for numerous dispersed applications, the necessity of deployment and maintenance of corporate network, and reduces training time. It is simple for end-users to utilize web-based solutions. In addition, a web-based architecture requires only lightweight software clients (i.e., web browsers).

Besides, there is a need for real-time data analysis, which induces memory and storage issues. Traditional OLAP (On line Analytical Processing) tools are often based on a cumbersome hardware and software architecture, so they require significant resources to provide a high performance. Their flexibility is limited by data aggregation. At the same time, in-memory databases provide significant performance improvements. Absence of disk I/O operations permits fast query response times. In-memory databases do not require indexes, recalculation and pre-aggregations, thus system becomes more flexible because analysis is possible to a detailed level without its pre-definition. Moreover, according to analyst firms, "by 2012, 70% of Global 1000 organizations will load detailed data into memory as the primary method to optimize BI application performance" [47].

Thus, our objective is to propose an original and adapted BI solutions for SMEs. To this aim, in a first step, we review in this paper the existing research related to this issue.

The remainder of this paper is organized as follows. In section 2, we first present and discuss web-based BI approaches, namely web data warehouses and web-based open source software for data warehousing. In Section 3, we review in-memory BI solutions (MOLAP, vector database-based BI software) and technologies that can support it (in-memory and vector databases). We finally conclude this paper in Section 4 and provide our view on how the research and technologies surveyed in this paper can be enhanced to fit SME's BI needs.

2. WEB-POWERED BI

The Web has become the platform of choice for the delivery of business applications for large-scaled entreprises as well as for SMEs. Web warehousing is a recent approach that merges data warehousing and business intelligence systems with web technologies [52]. In this section, we present and discuss web data warehousing approaches, their features, advantages and possibilities, as well as their necessity and potential for SMEs.

2.1 Web warehousing

2.1.1 General information

There are two basic definitions of web warehousing. The first one simply states that web warehouses use data from the Web. The second concentrates on the use of web technologies in data warehousing. We focus on second definition in our paper.

Web-data warehouses inherit a lot of characteristics from traditional data warehouses, including: data are organized around major subjects in the enterprise; information is aggregated and validated; data is represented by times series, not by current status. Web-based data warehouses nonetheless differ from traditional DWs. Web warehouses organize and manage the stored items, but do not collect them [52]. Web-based DW technology changes the pattern of users accessing to the DW: instead of accessing through a LAN (Local Area Network), users access via Internet/Intranet [30].

Specific issues raised by web-based DW include unrealistic user expectations, especially in terms of how much information they want to be able to access from the Web; security issues; technical implementation problems related to peak demand and load problems [42].

Eventually, web technologies make data warehouses and decision support systems friendlier to users. They are often used in data warehouses only to visualize information [18]. At the same time, web technology opens up multiple information formats, such as structured data, semi-structured data and unstructured data, to end-users. This gives a lot of possibilities to users, but also creates a problem known as data heterogeneity management [19].

Another important issue is the necessity to view the Web as an enormous source of business data, without whose enterprises loose a lot of possibilities. Owing to the Web, business analysts can access large external to enterprise information and then study competitor's movements by analyzing their web site content, can analyze customer preferences or emerging trends [11]. So, e-business technologies are expected to allow SMEs to gain capabilities that were once the preserve of their larger competitors [34]. However, most of the information in the Web is unstructured, heterogeneous and hence difficult to analyze [26].

Among web-technologies used in data warehousing, we can single out web browsers, web services and XML. Usage of web browser offers some advantages over traditional warehouse interface tools [19, 33]:

- cheapness and simplicity of web browser installation and use;
- reduction of system training time;
- elimination of problems posed by operating systems;
- low cost of deployment and maintenance;
- elimination of necessity for numerous dispersed applications;
- possibility to open data warehouse to business partners over an extranet.

Web warehouses can be divided into two classes: XML document warehouses and XML data warehouses. We present them in sections 2.1.2. and 2.1.3. respectively. We also introduce OLAP on XML data (XOLAP) in section 2.1.4. We finish this section by web-based paradigm known as cloud computing (section 2.1.5). Section 2.2. finally presents web-based open source software for data warehousing analysis.

2.1.2 XML document warehouses

An XML document warehouse is a software framework for analyzing, sharing and reusing unstructured data (texts, multimedia documents, etc.). Unstructured data processing takes an important place in enterprise life because unstructured data are larger in volume than structured data, are more difficult to analyze, and are an enormous source of raw information.
Representing unstructured or semi-structured data with traditional data models is very difficult. For example, relational models such as star and snowflake schemas are semantically poor for unstructured data. Thus, Nassis et al. utilize object-oriented concepts to develop a conceptual model for XML document warehouses [35]. They use UML diagrams to build hierarchical conceptual views. By combination of object oriented concepts and XML Schema, they build the xFACT repository.

2.1.3 XML data warehouses

In contrast to XML document warehouses, XML data warehouses focus on structured data. XML data warehouse design is possible from XML sources [3]. In this case, it is necessary to translate XML data into a relational schema by XML schema [3, 8]. Xyleme is one of the first projects aimed at XML data warehouse design [57]. It collects and archives web XML documents into a dynamic XML warehouse.
Some more recent approaches are based on classical warehouse schemas. Pokorny adapts the traditional star schema with explicit dimension hierarchies for XML environments by using Document Type Definition (DTD) [41]. Boussaïd et al. define data warehouse schemas via XML schema in a methodology named X-Warehousing [8]. Golfarelli proposes a semi-automatic approach for building the conceptual schema for a data mart starting directly from XML sources [15]. This work elaborates the concept of Dimensional Fact Model. Baril and Bellahsene propose a View Model from XML Documents implemented in the DAWAX (Data Warehouse for XML) system [4]. View specification mechanism allows filtering data to be stored. Nørvåg introduces a temporal XML data warehouses to query historical document versions and query changes between document versions [36]. Nørvåg et al. also propose TeXOR, a temporal XML database system built on top of an object-relational database system [37]. Finally, Zhang et al. propose an approach, named X-Warehouse, to materialize data warehouses based on frequent query patterns represented by Frequent Pattern Trees [58].

2.1.4 XOLAP

Some recent research attempts to perform OLAP analysis over XML data. In order to support OLAP queries and to be able to construct complex analytic queries, some researches extend the XQuery language with aggregation features [5].
Wiwatwattana et al. also introduce an XQuery cube operator, X^3 [55], Hachicha et al. also propose a similar operator, but based on TAX (Tree Algebra for XML)[17].

2.1.5 Cloud computing

Another, increasingly popular web-based solution is cloud computing. Cloud computing provides access to large amounts of data and computational resources through a variety of interfaces [38]. It is provided as services via cloud (Internet). These services delivered through data centers are accessible anywhere. Besides, they allow the rise of cloud analytics [2].
The main consumers of cloud computing are small enterprises and startups that do not have a legacy of IT investments to manage [50]. Cloud computing-based BI tools are rather cheap for small and middle-sized enterprises, because they provide no need of hardware and software maintenance [1] and their

prices increase according to required data storages. Contrariwise, cloud computing does not allow users to physically possess their data storage. It causes user dependence on the cloud computing provider, loss of data control and data security. In conclusion, most cloud computing-based BI tools do not fit enterprise requirements yet.

2.1.6 Discussion

Data storage and analysis interface solutions should be easily deployed in a small organization at low cost, and thus be based on web technologies such as XML and web services. Web warehousing is rather recent, but a popular direction that provides a lot of advantages, especially in data integration. Web-based tools provide light interface. Thereby, their usage by small and middle-sized enterprises is limited. Existing cloud-based BI tools are appropriated for small and middle-sized enterprises with respect to price and flexibility. However, they are so far enterprise-friendly and are in need of data security enhancements.

2.2 Web-based open source software

In this section, we focus on ETL(Extraction Transformation Loading) tools, OLAP servers and OLAP clients. Their characteristics are summarized in Table 1.

2.2.1 ETL

Web-based free ETL tools are in most cases ROLAP (Relational OLAP, discussed in Section 3.1.1.)-oriented. ROLAP-oriented ETL tools allow user to define and create data transformations in Java (JasperETL) or in TL (Clover.ETL)[1]. Singular MOLAP (Multidimensional OLAP, discussed in Section 3.1.1.)-oriented ETL Palo defines the ETL process either via web interfaces or via XML structures for experts. All studied ETL tools configure heterogeneous data sources and complex file formats. They interact with differents DBMSs (DataBase Management Systems). Some of the tools can also extract data from ERP (Enterprise Resource Planning) and CRM (Customer Relationship Management) systems [53].

2.2.2 OLAP

In this section we review OLAP servers as well as OLAP clients. All studied OLAP severs use the MDX (Multi-Dimensional eXpression) language for aggregating tables. They parse MDX into SQL to retrieve answers to dimensional queries. All reviewed OLAP servers exists for Java, but a Palo exists also for .NET, PHP, and C. Moreover, Palo is an in-memory Multidimensional OLAP database server[2]. Mondrian schemas are represented in XML files[3]. Mondrian Pentaho Sever is used by different OLAP clients, e.g., FreeAnalysis.
All studied OLAP clients are Java applications. They usually run on client, but tools also exist that run on web servers[53]. So far, only PocOLAP is a lightweight, open source OLAP solution.

2.2.3 Discussion

The industrial use of open source business intelligence tools is becoming increasingly common, but it is still not as wide- spread as for other types of software [53]. Moreover, freeware OLAP systems often propose simple web-based interfaces. In addition, there are some web-based open source BI tools that work in memory.
Nowadays, there are three complete solutions, including ETL and OLAP: Talend OpenStudio, Mondrian Pentaho and Pa- lo.
Among ETL tools, only Palo is MOLAP-oriented. Not all of these tools provide free graphical user interfaces. All three represented ETL tools support Java. They can be implemented on different platforms.
Free web-based OLAP servers are used by different OLAP clients. The most extended and widely used is Mondrian Pentaho Server due to its functionality. All studied OLAP clients are Java applications. Most of them can be used with XMLA(XML for Analysis)-enabled sources. But they have not enough documentation.
Generally, web-based studied tools provide sufficient functionality, but they remain cumbersome due to traditional OLAP usage.

3. IN-MEMORY BI SOLUTIONS

In the late eighties, main memory databases were researched by numerous authors. Thereafter, it has rarely been discussed because of limits of technologies at this time, but nowadays it takes back an important place in database technologies.

3.1 MOLAP

3.1.1 OLAP and MOLAP

Before studying existing MOLAP approaches, we review general OLAP principles and definitions. The OLAP concept was introduced in 1993 by Codd. OLAP is an approach to quickly answer multidimensional analytical queries [13]. In OLAP, a *di-*

[1]http://www.cloveretl.com

[2]http://www.jedox.com/en/products/palo_olap_server
[3]http://mondrian.pentaho.org/

		Tools	Platform	License	Particular features
ETL	ROLAP	Clover.ETL	Java	LGPL	does not have an open source GUI; uses its own TL language for data transformations
		JasperETL	Java	GPL	generated code - Java or Perl; can use CRM systems as data sources
	MOLAP	Palo ETL Server	Java	GPL	does not have a GUI for a while; parallel jobs are not supported
OLAP	servers	Mondrian	Java	CPL	ROLAP-based; data cubes via XML
		Palo	Java	GPL	MOLAP-based; works in memory; data cubes via Excel add-in
	clients	FreeAnalysis	Java	MPL	works with servers that use XMLA, e.g., Modrian
		JPalo	Java	GPL	works with the Palo server
		PocOLAP	Java	LGPL	

Table 1: Web-based open source software

mension is a sequence of analyzed parameter values. An important goal of multidimensional modeling is to use dimensions to provide as much context as possible for *facts* [21]. Combinations of dimension values define a cube's *cell*. A *cube* stores the result of different calculations and aggregations. There are three variants of OLAP: MOLAP, ROLAP, Hybrid OLAP (HOLAP). We compare these approaches in table 2.
With respect to ROLAP and HOLAP, MOLAP provides faster computation time and querying [48] due to a storage of all required data in the OLAP server. Moreover, it provides more space-efficient storage [40].
Since the purpose of MOLAP is to support decision making and management, data cubes must contain sufficient information to support decision making and reply to every user expectation. In this context, researches try to improve three main aspects: response time (by new aggregations algorithms [28], new operators [46]), query personalization, data analysis visualization [26].

3.1.2 Storage methods

Researchers interested in MOLAP focus a lot on storage techniques. In addition, most researches choose MOLAP as the most suitable among OLAP-techniques for storage [31], although MOLAP requires significant storage capacity. According to Kudryavcev, there are three basic types of storage methods: semantic, syntactical, approximate [23]. Syntactical approaches transform only data storage schemas. Semantic storage techniques transform cube structures. Approximate storage techniques compress initial data. One semantic storage technique is Quotient Cube. It consists in a semantic compression by partitioning the set of cells of a cube into equivalent classes, while keeping the cube's roll-up and drill-down semantics and lattice structure [25]. The main objective of such approximating storage technique such as Wavelets is range-sum query optimization [29]. In the syntactical approach DWARF, a cube is compressed by deleting redundant information [49]. Data are represented as graphs with keys and pointers in graphs nodes. Data redundancy decrease is provided by an addressing and data storage improvement.

3.1.3 Schema evolution

There are a lot of works that bring up the problem of schema evolution, because working only with the latest version hides the existence of information that may be critical for data analysis. It is possible to classify these studies into two groups: updating models (mapping data in the last version) and tracking history models (saving schema evolution). Other types of approaches look at the possibility for users to choose which presentation they want for query reponses. For instance, Body et al. proposed a novel temporal multidimensional model for supporting evolutions on multidimensional structures by introducing a set of temporal modes of presentation for dimensions in a star schema [6].

3.1.4 Discussion

Multidimensional OLAP is appropriate for decision making. It offers a number of advantages, including automatic aggregation, visual querying, and good query performance due to the use of pre-aggregation [39].Besides, MOLAP may be a good solution for the situations in which small to medium-sized DBs are the norm and application software

Table 2: Comparison of OLAP technologies

	MOLAP	ROLAP	HOLAP
Data storage	Multidimensional database	Relational database	Uses MOLAP technology to store higher-level summary data, a ROLAP system to store detailed data
Results sets	Stores in a MOLAP cube	Stores no results sets	Stores results sets, but not all
Capacity	Requires singificant capacity	Requires the least storage capacity	Compromise between performance, capacity, and permutations of dimensions available to a user
Performance	The fastest performance	The slowest performance	
Dimensions	Minimum number	Maximum number	
Vulnerability	Provides poor storage utilization, especially when the data set is sparse	Database design recommended by ER diagrams are inappropriate for decision support systems	
Advantages	Fast query performance; automated computation of higher level aggregates of the data; array model provides natural indexing	No limitation on data volume; leverage functionalities inherent in relational databases	Fast access at all levels of aggregation; compact aggregate storage; dynamically updated dimensions; easy aggregate maintenance
Disadvantages	Data redundancy; querying models with dimensions of high cardinality is difficult	Slow performance	Complexity - a HOLAP server must support MOLAP and ROLAP engines, tools to combine storage engines and operations. Functionality overlap - between storage and optimization techniques in ROLAP and MOLAP engines.

speed is critical [45], because loading all data to the multidimensional format does not require significant time nor disk space. Nevertheless, MOLAP systems have different problems due to the complexity, time-consuming and necessity of an expert for cube rebuilding. If the user wants to change dimensions, the whole deployment process need to be redone (datamart schema, ETL process, etc.) [56]. However, the cost of MOLAP tools does not fit the needs of small and middle-sized enterprises. In addition, MOLAP-based systems may encounter significant scalability problems. Moreover, MOLAP requires a cumbersome architecture, i.e., important software and hardware needs, the necessity of significant changes in work process to generate substantial benefits [32], and a considerable deployment time.

3.2 Main Memory Databases

3.2.1 General information

Main Memory Databases (MMDBs) entirely reside in main memory [14] and only use a disk subsystem for backup [16]. The concept of managing an entire database in main memory has been researched for over twenty years, and the benefits of such approaches have been well-understood in certain domains, such as telecommunications, security trading, applications handling high traffic of data, e.g., routers; real-time applications. However, it is only recently, with decreasing memory prices and the availability of 64-bit operating systems, that the size restrictions on in-memory databases have been removed and in-memory data management has become available for many applications [27, 54]. When the assumption of disk-residency is removed, complexity is dramatically reduced. The number of machine instructions drops, buffer pool management disappears, extra data copies are not needed, index pages shrink, and their structure is simplified. Design becomes simpler and more compact, and queries are executed faster [54]. Consequently, usage of main memory databases become advantageous in many cases: for hot data (frequently ac-

cess, low data volume), for cold data (scarce access, in the case of voluminous data), in application requiring a short access/response time.
A second wave of applications using MMDB is currently appearing, e.g., FastDB, Dali from AT&T Bell lab, TimesTen from Oracle. These systems are widely used in many applications such as HP intellect web flat already, Cisco VoIP call Proxy, the telecom system of Alcatel and Ericsson and so on [12]. The high demand of MMDBs is provoked by the necessity of high reliability, high real-time capacity, high quantity of information throughput [20].
MMDBs have some advantages, including short response time, good transaction throughputs. MMDBs also leverage the decreasing cost of main memory. Contrariwise, MMDB size is limited by size of RAM (Random Access Memory). Moreover, since data in main memory can be directly accessed by the processor, MMDBs suffer from data vulnerability, i.e., risk of data loss because of unintended accident due to software errors[14], hardware failure or other hazards.

3.2.2 MMDB issues

Although in-memory technologies provide high performance, scalability and flexibility to BI tools, they are still some open issues. MMDBs work in memory, therefore the main problems and challenges are recovery, commit processing, access methods and storage.
There is no doubt that backups of memory resident databa- ses must be maintained on other storage than main memory in order to insure data integrity. In order to protect against failures, it is necessary to have a backup copy and to keep a log of transaction activity [14]. In addition, recovery processing is usually the only MMDB component that deals with disk I/O, so it must be designed carefully [20]. Existing research works do not share a common view of this problem. Some authors propose to use a part of stable main memory to hold the log. It provides short response time, but it causes a problem when logs are large. So, it is used for the precommit transactions. Group commits (e.g., a casual commit protocol [27]) allow accumulating several transactions in memory before flushing them to the log disk. Nowadays, commit processing is especially important in distributed database systems because it is slow due to the fact that disk logging takes place at several sites [27].
Several different approaches of data storage exist for MMDBs. Initially, there have been a lot of attempts to use database partitioning techniques developed earlier for other types of databases. Gruenwald and Eich divide existing techniques as following: horizontal partitioning, group partitioning, single vertical partitioning, group vertical partitioning, and mixed partitioning [16]. Only horizontal and single vertical partitioning are suitable for MMDBs and, as a result of this study, single vertical partitioning was chosen as the most efficient [10]. B-trees and hashing are identified also as appropriate storage techniques for MMDBs. Hashing is not as space efficient as a tree, so it is rarely used [43]. Finally, most researches agree to choose T-trees (a balanced index tree data structure optimized for cases where both the index and the actual data are fully kept in memory) as the main storage technique [12, 14, 44]. T-trees indeed require less memory space and fewer CPU cycles than B-trees, so indexes are more economical.
Above-mentioned issues are important for BI environment: data coherence is strategic, performance is fundamental for on-line operations like OLAP. Choices of right storage and recovering techniques are crucial as it can damage data security and data integrity.

3.2.3 MMDB Systems

In this section we give an overview of MMDB systems. We particularly focus our discussion on the most recent systems such as Dali, FastDB, Kdb, IBM Cognos TM1 and TimesTen.
Among studied systems, we can distinguish a storage manager (the Dali system [20]) and complete main memory data- base systems (FastDB, Kdb, TM1, TimesTen). Interfaces can be based on zero-footprint Web (IBM Cognos TM1[4]), standard SQL (TimesTen)[54] or C++ (FastDB)[22]. Most MMDBS feature SQL or SQL-like query language (FastDB, TimesTen). Kdb system uses its own language "q" for programming and querying [24].
IBM Cognos 8 BI and TimesTen are aimed at decision making in large corporations. Main MMDB disadvantages are interprocess communication absence and high storage requirements (Dali system) [9], limitation of server memory (TimesTen), client-server architecture is unsupportable (FastDB).

3.2.4 Discussion

The main benefit of using MMDBs is short access/reponse time and good transaction throughput. But MMDBs are hampered by data vulnerability and security problems. Memory is not persistent, which means data loss in case of failure on the server. Security problems come from unauthorized

[4]www-01.ibm.com/software/data/cognos/products/tm1

access to data aimed at data corruption or theft.
So far, MMDBs are mainly used in real-time applications, telecommunications, but not commonly used for decision making.
In spite of a considerable research on MMDBs, there are some unresolved issues such as data security and safety and data processing efficiency.

3.3 Vector Databases

3.3.1 General information

A vector table is built by transforming a file in the following way: every record represents all column values in a vector. Vector databases (VDBs) do not require indexes nor any complex database structure. Differences between vector and relational databases are summarized in table 2. In order to access data, relational DBMSs provide only sequential scan by columns and by rows. VDBs provide fast data access. Besides, relational DBs store large volumes of repeatable data due to data nature. For example, in a table of students, French nationality can be repeated in a great number. Contrariwise, in VDBs, this data is present only once. It provides significant data compression.
The main principles of vector databases are data associations and data access by pointers. Vector database implementations allow elimination of data redundancy, because any possible pice of data is written once and it does not repeat itself. Such metadata as keys in the relational data model loose their interest in VDBs, because data associations are provided by pointers. Hence, VDBs do not consume as much space as relational DBs.

3.3.2 VDB-based BI

The main principle of vector database is that instead of dimension associations with OLAP cube there are associations between data. These associations are defined during data load process by matching up table columns having the same name. Usage of vector databases differs from classical warehousing: there is no predefinition of what a dimension is. Any piece of data is available as dimension and any piece of data is available as measure. So, it is not necessary to reconstruct data schema in the case of dimension change. As vector databases work in memory, VDB-based BI are endowed with instant data access. However, entreprises frequently hesitate to use VDB-based BI because of noninteroperability with SQL tools.
One BI tool that uses vector database deployment is QlikView[5]. QlikView provides integrated ETL. It removes the need to pre-aggregate data. It is possible to change analysis axes any moment at any level of query detailing. Despite QlikView capacities, it has some limitations and disadvantages such as lack of a unified metadata view and of predicting models (QlikView's statistical analysis features are less developed than the in other BI tools). There is no specialization in visualization: QlikView provides a clean interface to analysts but it lacks advanced visualization features to help them graphically wade through complicated data. One of the QlikViews's features is an ability to automatically connect tables. But this can create some problems. When there are fields, which represent the same thing in different tables and they do not have the same name, it is necessary to rename them to connect them. When there are fields in different tables that have the same name, but not the same content and sense, a senseless connection is created. So it is necessary to delete this connection and reanalyze all the fields with the same names in order to distinguish the ones with different sense. QlikView provides a possibility for end-users to use integrated ETL and to construct their data schema themselves, which often leads to unsatisfactory results.

[5]www.qlikview.com

3.3.3 Discussion

Vector databases hold the same advantages as others in-memory databases and are only limited by memory size.
VDB-based BI is a relatively new direction, but it is rather popular due to fast performance, great analysis capacity, unlimited number of dimensions, tables and measures and implementation easiness.
However, among features proposed by QlikView, there are disputable ones: automatic table connection, possibility to create a data schema by end-user. These characteristics do not cover different situations due to data aggregation complexity when data come from different sources. Such data have different refinement levels, different field names, etc. Consequently, providing to end-users the possibility to create data schemas can provoke an inadequate data schema, table connections, data loss as well as false data presence in database. Moreover, VDB-based BI tools are often blackboxes, meaning that we do not know what happens inside. Such models also lack flexibility.

4. CONCLUSION

Nowadays, BI becomes an essential part of any enterprise, even an SME. This necessity is caused by the increasing data volume indispensable for decision making. Existing solutions and tools are mostly

Table 3: Relational and vector database chatacteristics

Characteristic	Relational DB	Vector DB
Access to data	Sequential	Parallel
Data integrity	Foreign Keys	Multi-dimensional
Data relations stored in	Keys	Vectors
Data reuse	Not available	Built-in
Metadata	System tables	None
Speed (high volume)	Slow	Fast
Uniqueness	User Constraints	Built-in

aimed at large-scaled enterprises; thereby they are inaccessible or insufficient for SMEs because of high price, redundant functionality, complexity, and high hardware and software requirements. SMEs require solutions with light architectures that, moreover, are cheap and do not require additional hardware and software.

This survey discusses the importance of data warehousing for SMEs, presents the main characteristics and examples of web-based data warehousing, MOLAP systems and MMDBs. All these approaches have important disadvantages to be chosen as a unique decision support system: cumbersome architecture and complexity in MOLAP, data vulnerability in MMDBs, non-transparency and providing too large powers for users in VDB-based systems, security issues in cloud computing systems.

In this context, our research objective is to design BI solutions that are suitable for SMEs and avoid the aforementioned disadvantages.

Our idea is to work toward a ROLAP system that operates in-memory, i.e., to add in OLAP operators on top of an SQL-based MMDB. This should simplify a lot the in-memory OLAP architecture with respect to MOLAP. Choosing an open source MMDB system (such as FastDB) and using well-known ETL, modeling and analysis processes should also help avoid the "black box issue" of VDBs. Finally, storing business data as close to the user as possible mitigates security issues with respect to cloud BI. Problems will still remain, though (e.g., data vulnerability and need for backup, the design of adapted, in-memory indexes for OLAP), but we are confident we can address them in our future research.

5. ACKNOWLEDGMENTS

The authors would like to thank the French Ambassy in Ukraine for supporting this joint research work of the Kharkiv National University of Economics (Ukraine) and the University of Lyon 2 (France).

6. REFERENCES

[1] D. J. Abadi. Data Management in the Cloud: Limitations and Opportunities. *IEEE Data Engineering Bulletin*, 32(1):3–12, March 2009.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, February 2009.

[3] M. Banek, Z. Skocir, and B. Vrdoljak. Logical Design of Data Warehouses from XML . In *ConTEL '05: Proceedings of the 8th international conference on Telecommunications*, volume 1, pages 289–295, 2005.

[4] X. Baril and Z. Bellahsene. Designinig and Managing an XML Warehouse. In *XML Data Management: Native XML and XML-Enabled Database Systems*, chapter 16, pages 455–473. Addison-Wesley Professional, 2003.

[5] K. Beyer, D. Chamberlin, L. S. Colby, F. Özcan, H. Pirahesh, and Y. Xu. Extending XQuery for analytics. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 503–514, New York, NY, USA, 2005. ACM.

[6] M. Body, M. Miquel, Y. Bédard, and A. Tchounikine. A multidimensional and multiversion structure for OLAP applications. In *DOLAP '02: Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*, pages 1–6, New York, NY, USA, 2002. ACM.

[7] M. Böhnlein and A. U. vom Ende. *Business Process Oriented Development of Data Warehouse Structures*, pages 3–21. Physica: Heidelberg 2000, 2000.

[8] O. Boussaïd, R. BenMessaoud, R. Choquet, and S. Anthoard. X-Warehousing: an

XML-Based Approach for Warehousing Complex Data. In *ADBIS '06: Proceedings of the 10th East-European Conference on Advances in Databases and Information Systems*, volume 4152 of *Lecture Notes in Computer Science*, pages 39–54, Heidelberg, Germany, September 2006. Springer.

[9] P. Burte, B. Aleman-meza, D. B. Weatherly, R. Wu, S. Professor, and J. A. Miller. Transaction Management for a Main-Memory Database. *The 38th Annual Southeastern ACM Conference, Athens, Georgia*, pages 263–268, January 2001.

[10] Y. C. Cheng, L. Gruenwald, G. Ingels, and M. T. Thakkar. Evaluating Partitioning Techniques for Main Memory Database: Horizontal and Single Vertical. In *ICCI '93: Proceedings of the 5th International Conference on Computing and Information*, pages 570–574, Washington, DC, USA, 1993. IEEE Computer Society.

[11] W. Chung and H. Chen. Web-Based Business Intelligence Systems: A Review and Case Studies. In G. Adomavicius and A. Gupta, editors, *Business Computing*, volume 3, chapter 14, pages 373–396. Emerald Group Publishing, 2009.

[12] Y. Cui and D. Pi. SQLmmdb: An Embedded Main Memory Database Management System. *Information Technology Journal*, 6(6):872–878, 2007.

[13] S. C. E.F. Codd and C. Salley. Providing OLAP to User-Analysts: An IT Mandate, 1993.

[14] H. Garcia-Molina and K. Salem. Main memory database systems: An overview. *IEEE Transactions on Knowledge and Data Engineering*, 4:509–516, 1992.

[15] M. Golfarelli, S. Rizzi, and B. Vrdoljak. Data warehouse design from XML sources. In *DOLAP '01: Proceedings of the 4th ACM international workshop on Data warehousing and OLAP*, pages 40–47, New York, NY, USA, 2001. ACM.

[16] L. Gruenwald and M. H. Eich. Choosing the best storage technique for a main memory database system. In *JCIT '90: Proceedings of the 5th Jerusalem conference on Information technology*, pages 1–10, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[17] M. Hachicha, H. Mahboubi, and J. Darmont. Expressing OLAP operators with the TAX XML algebra. In *DataX-EDBT '08: 3rd International Workshop on Database Technologies for Handling XML Information on the Web*, pages 61–66, March 2008.

[18] H. A. A. Hafez and S. Kamel. Web-Based Data Warehouse in the Egyptian Cabinet Information and Decision Support Center. In R. Meredith, G. Shanks, D. Arnott, and S. Carlsson, editors, *DSS'04: The IFIP International Conference on Decision Support in an Uncertain and Complex World*, pages 402–409. Monash University, Australia (CD Rom), July 2004.

[19] C. Hsieh and B. Lin. Web-based data warehousing: current status and perspective. *The Journal of Computer Information Systems*, 43:1–8, January 2002.

[20] H. V. Jagadish, D. F. Lieuwen, R. Rastogi, A. Silberschatz, and S. Sudarshan. Dalí: A High Performance Main Memory Storage Manager. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 48–59, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[21] R. Kimball and M. Ross. *The Data Warehouse Toolkit: the complet guide to dimensional modeling*. Wiley Computer Publishing, 2002.

[22] K. Knizhnik. FastDB Main Memory Database Management System. Technical report, Research Computer Center of Moscow State University, Russia, March 1999.

[23] Y. Kudryavcev. Efficient algorithms for MOLAP data storage and query processing. In *SYRCoDIS '06: Proceedings of the 3rd Spring Colloquium for Young Researchers in Databases and Information Sytems*, page 5, Moscow, Russia, 2006.

[24] Kx Systems. The kdb+ Database White Paper. A unified database for streaming and historical data, 2009. Retrieved September 1, 2010 from http://kx.com/papers.

[25] L. V. S. Lakshmanan, J. Pei, and J. Han. Quotient cube: how to summarize the semantics of a data cube. In *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, pages 778–789. VLDB Endowment, 2002.

[26] G. Lawton. Users Take a Close Look at Visual Analytics. *Computer*, 42(2):19–22, 2009.

[27] I. Lee, H. Y. Yeom, and T. Park. A New Approach for Distributed Main Memory Database Systems: A Causal Commit Protocol. *IEICE Transactions on Information and Systems*, E87-D(1):196–204, January 2004.

[28] Y.-K. Lee, K.-Y. Whang, Y.-S. Moon, and

I.-Y. Song. An aggregation algorithm using a multidimensional file in multidimensional OLAP. *Information Sciences*, 152(1):121–138, June 2003.

[29] D. Lemire. Wavelet-Based Relative Prefix Sum Methods for Range Sum Queries in Data Cubes. In *CASCON '02: Proceedings of the 2002 conference of the Centre for Advanced Studies on Collaborative research*, page 6. IBM Press, October 2002.

[30] Z. Luo, Z. Kaisong, X. Hongxia, and Z. Kaipeng. The Data Warehouse Model Based on Web Service Technology. *Journal of Communication and Computer*, 2(1):26–31, January 2005.

[31] E. Malinowski and E. Zimányi. Hierarchies in a multidimensional model: from conceptual modeling to logical representation. *Data & Knowledge Engineering*, 59(2):348–377, 2006.

[32] M. McDonald. Light weight vs heavy weight technologies, the difference matters. Gartner, March 2010.

[33] A. Mehedintu, I. Buligiu, and C. Pirvu. Web-enabled Data Warehouse and Data Webhouse. *Revista Informatica Economica*, 1(45):96–102, 2008.

[34] R. Mullins, Y. Duan, D. Hamblin, P. Burrell, H. Jin, G. Jerzy, Z. Ewa, and B. Aleksander. A Web Based Intelligent Training System for SMEs. *The Electronic Journal of e-Learning*, 5:39–48, 2007.

[35] V. Nassis, W. Rahayu, R. Rajugan, and T. Dillon. Conceptual design of XML document warehouses. In *DaWak '04: Proceedings of the 6th International on Data Warehousing and Knowledge Discovery*, pages 1–14, 2004.

[36] K. Nørvåg. Temporal XML Data Warehouses: Challenges and Solutions. In *Proceedings of Workshop on Knowledge Foraging for Dynamic Networking of Communities and Economies(in conjunction with EurAsia-ICT'2002)*, October 2002.

[37] K. Nørvåg, M. Limstrand, and L. Myklebust. TeXOR: Temporal XML Database on an Object-Relational Database System. In M. Broy and A. V. Zamulin, editors, *Ershov Memorial Conference*, volume 2890 of *Lecture Notes in Computer Science*, pages 520–530. Springer, 2003.

[38] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The Eucalyptus Open-Source Cloud-Computing System. In *CCGRID '09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 124–131, Washington, DC, USA, 2009. IEEE Computer Society.

[39] D. Pedersen, K. Riis, and T. B. Pedersen. XML-Extended OLAP Querying. In *SSDBM '02: Proceedings of the 14th International Conference on Scientific and Statistical Database Management*, pages 195–206, Washington, DC, USA, 2002. IEEE Computer Society.

[40] T. B. Pedersen and C. S. Jensen. Multidimensional Database Technology. *Computer*, 34(12):40–46, 2001.

[41] J. Pokorný. XML Data Warehouse: Modelling and Querying. In *Baltic DB&IS '02: Proceedings of the 5th International Baltic Conference on Databases and Information Systems*, pages 267–280. Institute of Cybernetics at Tallin Technical University, 2002.

[42] D. J. Power and S. Kaparthi. Building Web-based Decision Support Systems. *Studies in Informatics and Control*, 11(4):291–302, December 2002.

[43] J. Rao and K. A. Ross. Cache Conscious Indexing for Decision-Support in Main Memory. In *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*, pages 78–89, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[44] R. Rastogi, S. Seshadri, P. Bohannon, D. W. Leinbaugh, A. Silberschatz, and S. Sudarshan. Logical and Physical Versioning in Main Memory Databases. In *VLDB '97: Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 86–95, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[45] P. Rob and C. Coronel. *Database systems: design, implementation, and management*. Cengage Learning, 2007.

[46] G. Sathe and S. Sarawagi. Intelligent Rollups in Multidimensional OLAP Data. In *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, pages 531–540, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[47] K. Schlegel, M. A. Beyer, A. Bitterer, and B. Hostmann. BI Applications Benefit From In-Memory Technology Improvements. Gartner, October 2006.

[48] F. Silvers. *Building and Maintaining a Data*

Warehouse. Auerbach Publications, 2008.

[49] Y. Sismanis, A. Deligiannakis, N. Roussopoulos, and Y. Kotidis. DWARF: shrinking the PetaCube. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 464–475, New York, NY, USA, 2002. ACM.

[50] J. Staten. Is cloud computing ready for the enterprise? Forrester Research, March 2008. Retrieved September 1, 2010 from http://www.forrester.com/rb/Research/is_cloud_computing_ready_for_enterprise/q/id/44229/t/2.

[51] M. Stonebracker and N. Hachem. The End of an Architectual Era (It's Time for a Complete Rewrite). In *VLDB'07: Proceedings of the 33rd international conference on Very Large Data Bases*, pages 1150–1160, 2007.

[52] X. Tan, D. C. Yen, and X. Fang. Web warehousing: Web technology meets data warehousing. *Technology in Society*, 25:131–148, January 2003.

[53] C. Thomsen and T. B. Pedersen. A Survey of Open Source Tools for Business Intelligence. *International Journal of Data Warehousing and Mining*, 5(3):56–75, jul-sep 2009.

[54] C. TimesTen Team. In-memory data management for consumer transactions the timesten approach. *SIGMOD Record*, 28(2):528–529, 1999.

[55] N. Wiwatwattana, H. V. Jagadish, L. V. S. Lakshmanan, and D. Srivastava. Xˆ3: A Cube Operator for XML OLAP. *IEEE 23rd International Conference on Data Engineering*, pages 916–925, 2007.

[56] G. Xie, Y. Yang, S. Liu, Z. Qiu, Y. Pan, and X. Zhou. EIAW: Towards a Business-friendly Data Warehouse Using Semantic Web Technologies. In K. Aberer, K.-S. Choi, N. Noy, D. Allemang, K.-I. Lee, L. J. B. Nixon, J. Golbeck, P. Mika, D. Maynard, G. Schreiber, and P. Cudre-Mauroux, editors, *ISWC/ASWC '07: Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference*, volume 4825 of *LNCS*, pages 851–904, Berlin, Heidelberg, November 2007. Springer Verlag.

[57] L. Xyleme. A dynamic warehouse for XML data of the Web. *IEEE Data Engineering Bulletin*, 24:40–47, 2001.

[58] J. Zhang, W. Wang, H. Liu, and S. Zhang. X-warehouse: building query pattern-driven data. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 896–897, New York, NY, USA, 2005. ACM.

Elisa Bertino Speaks Out

on How She Accrued 301 Coauthors, Revitalized a Department, Cut Her Commute to Three Minutes, Enhanced Our Trust in Shared Data, and More

by Marianne Winslett

http://homes.cerias.purdue.edu/~bertino/

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 we are in West Lafayette, Indiana. I have here with me Elisa Bertino, who is a professor of computer science at Purdue University and the Research Director of CERIAS, which is Purdue's center for security research. Before joining Purdue, Elisa was on the faculty at the University of Milan for many years. Elisa's research interests lie in database systems and information security. She is a former co-editor in chief of the VLDB Journal, *and is on the editorial boards of* ACM Transactions on Information and System Security, IEEE Internet Computing, *and* IEEE Security & Privacy. *Elisa's PhD is from the University of Pisa. So, Elisa, welcome!*

Elisa, you have written a ***lot*** *of papers and you are listed very high in CiteSeer. For example, you have 25 papers listed in DBLP just for 2007, and you have 301 coauthors in DBLP. How do you collaborate with so many people?*

I really like to work with other people a lot. I try very much to understand what other people are working on, because I feel that I learn a lot from them, more than they learn from me. This goes back to my very early experience, when I was a very young researcher, 23 years old. At that time, I had a lot of ideas for research that I wanted to do, but I was always running into some very senior people in

our field who dismissed all my ideas. Later, I discovered that some of those ideas of mine had not been wrong.

Because of that experience, I really try to listen to the ideas of other people and encourage them to pursue their ideas. In many cases, I end up collaborating with them. And I really feel that I can understand research in other fields in computer science as applied to databases or security. So that's why I have all those collaborations. Also, I have been working for many years – that explains that number of coauthors!

Well, maybe! I don't think I will ever have 301 coauthors! When you advise your PhD students, are you mainly listening?

I think that each person is unique, so there is no one single approach which is good for everyone. Some students have their own ideas and a lot of enthusiasm, so when I see that, I really try to encourage them. Of course, I try to make sure that they don't take a really wrong direction, but I also really try to understand what could be of value in what they propose. Other students may really need a lot of guidance, because some are shy and don't want to express their own ideas. Sometimes I give the student a problem when I see that the student does not have his own ideas, or is too shy.

I have heard that you became the department head at the University of Milan when you were very young, and then worked to make the department more research-oriented. For those of our readers who want to build up a research group or a department: what are the steps that you take to do that?

In Milan, I wanted to invest in people, especially younger people. A major problem in Italian universities is that the faculty are quite old, and we don't have enough young people. So I tried to use the resources to get younger people, to get postdoctoral fellowships, and so on. In some cases, I said that I didn't want to buy more equipment, I wanted to get people instead. That was a big key.

Once you have people, the next step is for the people to have interesting research ideas and collaborate, even across the disciplines. That goes back to my idea that I really like a lot to do research. I think that *everyone* should like to do research, so I usually do my best to make other people do research, interact, and come up with exciting research ideas.

How did you have time to do research and be the department head?

That was very tough, to be honest. In Italy, I had to be the head of a department with a faculty of about 50 professors, and also I had to teach. One year I was teaching 20 hours a week, so I was working hard!

Fortunately, unlike US universities, Italian universities hire their own former students, which gives research groups a lot of stability. (But it has other disadvantages, I must say.) When I was the department head, my research group had several assistant professors, and some of them had been my own students. This stability helps a lot, because the older ones can help with the younger students and so forth. But there was still a big stress, I must say!

Recently some well-known database researchers have moved from the US to Europe, but you have bucked this trend by moving from Italy to the US. What convinced you to move?

There are several reasons. First of all, even though I was a professor in Italy, I had spent periods of time in the US here, visiting various institutions. In the back of my mind, I had the idea that I wanted to come to really see how it is to be a professor in the US. I had visited Purdue about six years ago, so

I knew several people there, and I knew that the Midwest has really strong research groups in databases and in security. When an opportunity was available for a faculty position from Purdue, they asked me if I had an interest in considering a position here. At that moment, it was something I really wanted to do, and that I had wanted to do for many years. I thought that it could be an opportunity that I shouldn't miss. It was a tough decision, since I had built a lot in Italy, but I also saw it as an opportunity to improve my research, by being able to work with the really strong research groups in the Midwest. You have to follow your instincts sometimes, and that's what I did.

What do you miss most about Italy?

That's an interesting question! What I would miss in the past, but not today in the Internet age, was not to know what was happening in Italy. When I was living in San Jose many years ago, you wouldn't be able to get an Italian newspaper unless you would drive up to San Francisco. But today, with the Internet, this is not an issue. You can get TV online, and you can be really up to date on whatever good and bad happens in Italy. Sometimes I miss a little bit the fact that in Italy you can easily go to the sea, to the mountains, and I miss a little bit the food. But other than that, I don't miss much, because you can get almost everything today, the world is so connected.

Wasn't it strange to move from such a big cosmopolitan city like Milan to Lafayette, which is a fairly small and quiet place?

This is a question that everyone asks me! Actually, I still feel like I am on vacation being here. Milano is a nice city because it is cosmopolitan, but it is also very stressful, with a lot of traffic jams. I had to get up very early, at 6 o'clock, to get to my office. You don't meet your friends there very easily because you have to drive two hours. And here, everything is so close, so comfortable; there is much more social interaction. And it may seem minor, but in Milan, there is an excess of noise, and you cannot see the stars because the sky has always some pollution. Here sometimes there is a perfect silence, which I really like a lot.

I think life goes in cycles. Maybe in 10 years when I'll be tired of being in a quiet place, I'll go somewhere else!

How long is your drive to work now?

Oh, it's maybe three minutes.

Mine's four minutes, so you have it better!

Would you recommend that new computer science PhD graduates in the US consider a job in Europe?

I think that working in Europe could be an important experience for young people. There are very good institutions there today. Universities care much more about ranking than they used to before, so there is much more competition there now. Some countries in Europe are very good at computer science and they are trying to attract a lot of talented young PhD researchers. The European Union is trying its best to help people moving from one country to another.

Of course, there are differences among different countries. In Italy, you do need to speak the language, because otherwise you wouldn't be able to teach, except for very few places. The funding may also be different depending on the country. Typically, each European country has its own national funding. The national funding can be very low in some countries, and then you have to work to get funding from the European Union. I feel that getting EU funding is much more difficult than getting funding in the US, because you have to write big proposals with a lot of countries.

Participating in those proposals is very interesting, and you get exposed to different cultures.

You were born in Rome, which at that time did not have a tradition of work in computer science. What led you to get into computer science in general, and databases and security in particular?

Both my parents have doctorates in mathematics, so I saw mathematical symbols all the time. I wanted to study mathematics since I was three years old! But at that time in Italy, if you studied mathematics, you would end up being a high school teacher. My mother thought that was not an exciting career. She said that I needed to study computer science instead, and I followed her advice.

Now, about databases, how did I get into that? I studied in Pisa, and at that time the professors there were very theoretical, doing work on formal semantics, programming languages, or complexity theory. I tried to find an advisor, but they were all busy with other students. So I ended up working on a thesis on databases with someone in a research institution in Pisa.

When I went to IBM in San Jose as a postdoc, I was working on database views for security. Once I started to work on database security, I learned more and more things about security. I got into other security topics, like digital identity management. I keep going back and forth between databases and security, in a way.

You were a postdoc at IBM Almaden, and I have heard that there are some interesting stories from your time there.

That was a very exciting time. I was just out of school from Pisa when I joined the System R* group. There were people there like Bruce Lindsay, Laura Haas, Pat Selinger (she was the manager of my group), Paul Wilms, and Guy Lohman. There were so many good people that I would walk around, looking at the names on the doors, and I couldn't believe my eyes to be there! It was very exciting! I really learned a lot from them. I had to do a lot of programming, which really made me learn the inside of a DBMS.

And the people were all very young. I remember working with Laura Haas, who had just started also with IBM. One day we were trying to debug some code and we were looking at the value stored in a particular location in memory. Before a certain instruction executed, there was one value in that location, and afterwards there was another value there, and we couldn't understand why. We tried everything we could think of, and then in the end, we went to talk to Bob Yost, who was the system manager there. We told him that the IBM 370 must have some hardware problem! Bob told us that this is not possible, go back and look again... then finally we found out that the problem was in the way the compiler would translate the move character instruction. Once we realized this, then we fixed the problem.

After working at IBM Almaden and MCC, you left industry. What makes you prefer academia to industry?

I feel that in academia, you are free to do what you like. You are your own boss. You can really decide your own research directions. And also, in Italy, we don't have industrial research labs where you can go to work. There is nothing there like an IBM research lab, or a Microsoft research lab. So if you are in industry in Italy, either you work for consulting companies, or you do applications.

What impact do you think September 11 had on security research?

Of course, that attack has shown that there are terrorist groups who will do what they can do to harm a country, and therefore, protecting cyber infrastructure is a key component of an overall security solution. But also, the attack has shown the importance of databases for security. After September

11, people understood that from data you can get a lot of useful information that can really be used to try to prevent those types of attacks, and to manage emergencies. People understood that all this large amount of data needed to be integrated, to be analyzed, so that you can extract useful knowledge. You need to be able to make relevant information available very quickly, so people can have it whenever they need it – but without being flooded with too much data, which is the other really big problem!

What do you view as the most important problems in information security today?

At the end, I think that the really important security problem is securing data and being able to trust the data. More and more people provide content and share data. However, can we really trust this data? Can we really rely on this information? Many years ago, the European Union started a research program to provide better and higher quality information on the Web. They observed that in a lot of cases, people go on the web to find medical information, and then believe this information. How can we make sure that this information is good? A lot of people use Wikipedia now; can you trust Wikipedia? And we will have the same problem for data in databases, which are more protected. They say that your network can go down, or your machine can be broken, but as long as your data is safe, you are okay. I am really interested about how we can assign trust levels to information.

So what approach do you think we could take to solve this information quality and information trust problem?

It is a really interesting research challenge. I would say that you have to combine a lot of different solutions. For example, semantic integrity is a very well known approach we have in databases, but it may not be enough. You may also need to use approaches to improve the quality of the data. You may need to keep track of provenance information, and in fact there has been a lot of work done in this area. But somehow this type of work has never considered the possibility that you may have malicious parties who may try to change the provenance information, and then may try to deceive you by providing the wrong data.

My group has been examining approaches where we look at multiple sources that supply information on the same topic. We decide how similar the information is that they provide, and determine the distance between them. Based on that, we can rate the trustworthiness of a party that provides information. We also need to consider that somebody will be using this information to make a decision. There can be policies for deciding whether to use a certain data item in making a decision, based on its trustworthiness. In the end, it is better to tell a user, e.g., that this data we couldn't verify and it's not trusted, so you may not want to use it if you have to make a critical decision. Then the user will be aware that the data is not so trusted, and can decide whether to use the data or not. There can also be policies for deciding whether you want to admit certain data in your databases, based on how trusted it is.

Do you see differences between the database and security research communities, for example, in the way program committees function or the way work is evaluated?

I have been in the program committee of the Oakland conference (the IEEE Symposium on Security and Privacy) and CCS (the ACM Conference on Computer and Communications Security). Oakland and CCS are both very selective conferences, but they receive fewer submissions than we do in the database conferences. Oakland usually has an in-person PC meeting.

A very interesting difference is that a lot of security research looks at attacks and defense, and we don't do that in databases. Many database security papers don't have a security analysis, where you show the possible threats to your approach or techniques, and possible mitigation. We always have a security analysis in papers we send to security conferences. While the type of research I do is more

on policy-based access control, I think it is also fun doing this type of research on attacks and defense. Sometimes I wonder whether the emphasis on attacks and defense is a US attitude, because the US had its Wild West, the frontier to conquer. Perhaps now there is the idea that you are the West, reasoning in terms of enemies trying to attack you.

Another difference is that in databases, we give a lot of attention to performance; performance is the key. Some security papers talk about performance, but usually this is not the main aim of security research. The main emphasis is to make sure that you analyze the security of your technique and your system.

Traditionally, it has been hard for professors to collaborate with researchers in industry, due to intellectual property (IP) issues. Can you tell us about your new Open Collaborative Research (OCR) project with IBM? (See http://www.cs.purdue.edu/news/12-14-06IBMocr.htm .)

The goal of IBM's program was to try to develop common research but make the research results and artifacts such as software available in the open source community. And therefore, we didn't really have IP rights to consider. Purdue was very open minded and agreed to these terms.

Our team's collaboration is going very well. Our project includes Purdue and IBM, and CMU as well. CMU is doing interesting work on usability for security and privacy techniques. So we are getting to learn new things, while functioning as one team.

Are other companies doing that, or is it just IBM?

I think that the OCR program grew out of an agreement that was signed by several companies, including HP, IBM and others. These companies noticed that IP is getting in the way of doing good joint research which would be a benefit to companies and universities. I don't know about the other companies, but IBM really tried to implement this agreement, by starting the OCR program.

What have you learned from your PhD students?

They really provide me with a lot of enthusiasm to continue my research. It is because of them that I never get tired. It is really exciting to see their ideas, and discuss with them. It is really a learning process.

It is also very interesting is to observe that people are very different. For example, I am proud that I have had a lot of female PhD students. Sometimes they are different from the male students, perhaps more shy but more dedicated to their work in a way. But again, as I said before, each person is different from each other.

How do you make time for everything?

I do as everyone else does. At least here at Purdue, everyone works a lot. Even after so many years, I really like research, so I don't feel like I am working. As a consequence, I work a lot. And so this is how I manage to do many things, I guess.

Do you have any words of advice for fledgling or mid-career database researchers or practitioners?

I think today we live in an exciting time. Computer scientists are changing the modern world. Probably we never anticipated that our field could have such an impact. I think about the Web 2.0, the new Web 3.0, there are so many applications!

I feel it is really important to have a vision and keep track of the trends in the world. I think that you

have to think about the future, because there are unprecedented opportunities and possibilities. Thinking about that makes you realize that you are really doing something exciting.

Among all your past research, do you have a favorite piece of work?

I have one nice piece of work on query optimization with unstructured data [Elisa Bertino, Fausto

Rabitti, and Simon J. Gibbs. Query Processing in a Multimedia Document System. *ACM Transactions on Information Systems* 6(1): 1-41 (1988)]. In 1988, nobody was working on unstructured data at all. I still remember that I was thinking to send the paper to SIGMOD, but then I decided not to. I was sure my paper would be rejected. The PC members would say, what is this "unstructured data"!

The paper is about a query optimizer for a multimedia document management system which had text and structured fields. We had to define a query language which had the equivalent of XPath expressions, and I had to develop the optimizer for it. Fortunately, I had learned a lot at IBM, so I implemented all of the optimizer and wrote my first ACM Transactions paper. I really liked that work.

Right now, in a lot of cases, the software produced by my research projects is developed by my students. At that time, I did everything by myself: the theory, the idea, the organization, the implementation, the testing, everything.

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

I would try to become a *cordon bleu* chef. I like cooking a lot! Especially in Milan you get spoiled. Milan has a lot of restaurants, and very good places from which you can take away food. For ten years, I didn't cook at all. But now here at Purdue, I do cook a lot, which I like anyhow. I like to improve.

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

I would like to do hardware-oriented research. Dennis Tsichritzis gave me a very important piece of advice more than 20 years ago. He told me that if you want to understand where new research in software has to go, look at the technological advances in hardware. Those advances will really tell you where you have to direct your research. I think that nanotechnology is really an exciting field. I wish I could have done a lot of work in hardware.

Thank you very much for talking with me today.

Thank you, Marianne. I really enjoyed our conversation.

The Chair's Report on SIGMOD'10 Demonstration Program

Yufei Tao
Chinese University of Hong Kong
taoyf@cse.cuhk.edu.hk

1. STATISTICS AND FACTS

Program committee. The PC team[1] consisted of 4 and 8 reviewers from research labs and universities, respectively. Geographically, 7 reviewers were from North America, 3 from Europe and 2 from Asia.

Submission and review process. We had a total of 95 submissions, up from 86 last year. Each submission had no more than 4 pages, one more than last year. Unlike research-paper submissions (which were subject to double-blind reviewing), each demo paper had the full author list exposed. This tradition, which was followed in all the preceding SIGMOD conferences, is mainly because, in general, it is difficult, sometime even awkward, to achieve anonymity of systems, many of which are long-term projects with a homepage, and thus, is easily identifiable with Google.

The review process was quite standard. Each submission was refereed by three reviewers. Papers with unanimous positive (negative) ratings from all reviewers were immediately accepted (rejected). The rest (i.e., those with conflicts) then went through a discussion phase. The competition turned out to be rather heated. The high quality of the submissions rendered rejection a painful decision to make. We therefore decided not to limit the length of the acceptance list, but accept a paper as long as (i) its average rating was at least "weak accept", and (ii) it was championed by at least one reviewer. Eventually, 35 demos were accepted, giving an acceptance rate of 36.8%.

Demo program. The 35 accepted papers were grouped into 4 groups, each having 8 or 9 demos. At the conference, every group was allowed two sessions of 90 minutes (i.e., 3 hours in total) for demonstration to the conference attendees. Each group was provided with a table for equipment setup and an easel to support a poster. All sessions proceeded smoothly with a decent amount of visitor traffic.

[1]The full name list can be found at *http://www.sigmod2010.org/org_sigmod_demo_pc.shtml*.

2. BEST DEMO AWARD

The best-demo award has not been a tradition of SIGMOD. It was introduced for the first time in SIGMOD'05, but was suspended during 2006-2008, and only resumed last year. This year the award was given for the 3rd time. There seems to have been consensus that it is going to become a regular award.

Differences from the best-paper award. Deciding the winner of the best-demo award can be even more difficult than the best-paper counterpart. The main difficulty comes from the fact that the decision cannot be made by reading only the demo papers. Remember that a demo paper is short (4 pages this year). Furthermore, it does not help much even if they are allowed to be much longer, because the referees must actually see the system to get an accurate feeling about its power, user-friendliness, scalability, etc. This, in turn, immediately rules out the possibility of concluding the winner before the conference (as is possible, and usually practised, for best-paper awards).

The necessity of on-site evaluation creates more issues. The most serious of all is to find enough referees. Keep in mind that it is rather time consuming to assess a demo, because enough time must be allocated to enable a referee to weight the strengths and weaknesses of a system accurately. A rule of thumb is that the time of evaluating a demo should be comparable to the duration of a talk in a research session. Another fact to be taken into account is that a SIGMOD attendee usually considers it more important to attend research talks than seeing a demo. In other words, it is unrealistic to expect that a referee would agree to evaluate too many demos. On the other hand, to allow better global consistency in the evaluation, it is a good idea to have at least some dedicated referees that will inspect a large number

of the demos. These referees deserve respects, because they most likely are skipping many research talks that they would like to go to originally.

The second main issue is time. Besides the considerable amount of time needed for demo assessment, sufficient time must also be allocated for post-evaluation discussion, especially when several demos have close scores without an obvious tie-breaking solution. The time problem is particularly acute, if the demo chair hopes to announce the award at the award ceremony, which often takes place at slightly beyond the halfway of the conference. In that case, each demo may have only a single chance to be evaluated, because its second appearance would very probably be scheduled in the second half of the conference.

Referees. This year, I recruited two types of referees. A *global referee* is required to evaluate (at least) half of all the demos, whereas a *local referee* needs to assess 4-5 demos of one group (recall that each group has 8-9 demos, so a local referee effectively sees half of a group). A local referee needs to spend 90 minutes (i.e., one session), which appears to be a reasonable commitment. A global referee, however, is required to stay for 4 sessions, or totally, 6 hours.

There were 3 global referees: Chris Mayfield (Purdue Univ. USA), Andy Pavlo (Brown Univ. USA), and myself. The total number of local reviewers[2] was 17.

Evaluation process. All the demos were considered for the award according to the following 3 criteria:

- *System completeness.* We preferred full-fledged systems to half-baked ones. After all, the best demo paper should not be merely a "long version" of the experimental section of a research paper.
- *Problem importance.* Obviously, systems dealing with novel and urgent problems deserve additional credits.
- *Technical depth.* The award-winning system should have overcome certain major technical obstacles. This criterion rules out the "labor-intensive" systems built on either existing or straightforward techniques.

Each demo was assigned at least 1 global referee and 2 local referees. The assessment involved a *team phase* and, if time permitted, an *individual phase*. In the team phase, all referees assigned to a demo formed a team. The authors were given 18 minutes to impress the team, including up to 12 minutes of introduction to their system, followed by (at least) 6 minutes of questioning-asking by the referees. The value 18 was determined so that a referee who needed to evaluate 5 demos was able to do so within the 90-minute timeframe of a session. In the voluntary phase, the referees acted individually, and had the freedom of learning more about any demos of her/his choices. At the end of evaluation, the referees convened for a short discussion, and then scored the demo separately based on the 3 criteria mentioned earlier. The demo's final score is the average of the scores by all the referees. Finally, the global referees in their last (long) meeting ranked all the demos, and decided the winners.

All of the above information, including the names of the referees assigned to each demo, the assessment criteria, and the time allocation, etc., was announced to the demo authors well before the conference, and publicized online again well in advance.

Winners. The best-demo award eventually went to

A tool for configuring and visualizing database parameters,
by Vamsidhar Thummala, and Shivnath Babu,
from Duke University, USA.

We also decided to give honorable mentions to:

1. *DCUBE: Discrimination discovery in databases*, by Salvatore Ruggieri, Dino Pedreschi, and Franco Turini, from Universita di Pisa, Italy.

2. *K*SQL: A unifying engine for sequence patterns and XML*, by Barzan Mozafari, Kai Zeng, and Carlo Zaniolo, from UCLA, USA.

3. FOR THE FUTURE

Below are two issues that my experience in the past year has left with me. Unfortunately, they still confound me at this time. I hope the future demo chairs will be able to find clear answers, and by doing so, enhance the quality of their demo programs.

- *Systems before submission?* It came to my attention that, for some accepted demos, implementation was done after the papers were submitted. In other words, what was written in those papers was essentially in the authors' vision, as opposed to the real stuff. Should this be allowed?

[2]Name list on *http://www.cse.cuhk.edu.hk/∼taoyf/sigmod10bd.html*.

I exchanged some emails with Divy Agrawal (SIGMOD'10 PC chair of the research program) about this. Eventually, we decided to follow the policy of the past SIGMODs, which is not to discriminate against such submissions. After all, it is possible that, at the time of writing the paper, the authors came up with some nice ideas that they did not have time to implement yet, but they knew they would be able to accomplish. Not much harm could be done if they decided to write about these ideas in the paper first, just to meet the paper deadline. In any case, it appears to be rather difficult to clearly define how much implementation is appropriate at the time of submission, not to mention that it will be technically challenging for the committee to verify the degree of implementation during the review process.

On the negative side, one can argue that the absence of any implementation requirement makes it possible for some authors to say whatever they want at submission time to stand an unfairly high acceptance chance, and then implement only part of what they claim after the paper has been accepted. It is unlikely that anyone can find out at the conference anyway. Furthermore, even disregarding such a conspiracy theory, we should probably be reminded that a demo paper, by definition, ought to describe a demo — but how do you know what to demonstrate without even the system?

- *Video submission?* In SIGMOD'09, each demo submission could optionally be accompanied by a video submission. The idea also generated some interesting debates. One, in particular, revolves around the fact that the extra efforts of preparing a video actually do not necessarily increase the acceptance chance. This is in fact not too surprising. Without any video, a reviewer can only imagine the system. As dreams are often pretty, the reviewer may be picturing something fancier than the actual system, and thus, generously give a high score. A video therefore has the disadvantage of potentially smashing the beautiful imagination, and hence, incurring a lower score.

  As a side note, video submissions, when enforced in a compulsory manner, appear to be an effective weapon to discourage implementation after submissions.

The SIGMOD 2010 Programming Contest A Distributed Query Engine

Clément Genzmer [1] Volker Hudlet [2] Hyunjung Park [3]
Daniel Schall [2] Pierre Senellart [4]

[1] Facebook, USA [2] TU Kaiserslautern, Germany
[3] Stanford University, USA [4] Télécom ParisTech, France

ABSTRACT

We report on the second annual ACM SIGMOD programming contest, which consisted in building an efficient distributed query engine on top of an in-memory index. This article is co-authored by the organizers of the competition (Clément Genzmer, Pierre Senellart) and the students who built the two leading implementations (Volker Hudlet, Hyunjung Park, Daniel Schall).

1. CONTEXT

For the second year in a row, a programming contest was organized in parallel with the ACM SIGMOD 2010 conference. Undergraduate and graduate student teams from over the world were invited to compete to develop an efficient distributed query engine over relational data. Students had several months to work on their implementation, which was judged for their overall performance on a variety of workloads. The teams responsible for the five best systems were invited to present their work during the SIGMOD 2010 conference, and the winning team (one-man team *cardinality* formed of Hyunjung Park, Stanford University), was awarded a prize of $5,000.

In addition to encouraging students to be active in the database research community, the aim is to build over the years, throughout a series of contests, an open source in-memory distributed database management system. Thus, the candidates of this year's contest relied on the in-memory index implementation produced as the outcome of last year's competition.

We first describe in more detail the task the contestants were involved in, as well as the workload their implementation was evaluated on. We then report on the outcome of the competition, before describing the key ideas of the systems ranked first and second.

2. TASK DESCRIPTION

As previously mentioned, the task was to program a simple distributed relational query engine. Contestants had to provide a binary library conforming to a specific interface, along with the corresponding source code. Each submission was evaluated on a dedicated cluster of eight machines, over a series of eight secret query loads. The input provided to the implementation for each workload was the description of which nodes of the cluster stored (parts of) which tables, possibly horizontally partitioned, as well as a set of queries, expressed in a simple subset of SQL. The goal was then to provide the correct output to these queries, as fast as possible. The final score of each submission was computed as a monotonous function of the total time used for running all workloads. Workloads where the submission crashed, did not return the correct output, or ran over the time limit of five to ten minutes (depending on the workload), were assigned penalties.

All queries were simple select-project-join queries, of the form:

```
SELECT alias.attribute, ...
FROM table AS alias, ...
WHERE condition1 AND ... AND conditionN
```

where a condition might be any of:

- `alias.attribute = constant`
- `alias.attribute > constant`
- `alias1.attribute1 = alias2.attribute2`

A parser for this subset of SQL was provided.

Attribute values were either character strings or integers, and tables were stored in text files on disk. All tables had at least one column indexed in memory, the implementation of the index being provided based on last year's contest. Before the actual starting of each workload, the contestants were given a predefined number of seconds to perform some preprocessing steps over the data. At this point, their implementation received a set of queries which was representative of the workload.

Among the eight benchmarks, five were of a reasonably small scale and were designed to test the workability of the binary provided by the contestants. The last three were designed to test the performance (up to 150,000

queries, and up to 1,000,000 tuples stored on a given node). Contestants were given a one-line description of each workload, though the actual structure of the input data was not disclosed. The benchmarking tool would initiate 50 parallel connections on a master node and then would start issuing the queries.

The full description of the task is available at `http://dbweb.enst.fr/events/sigmod10contest/`.

3. THE COMPETITION

The initial description of the task was made available in December 2009 along with all necessary interfacing code, though some addition and fixes came over the following months as bugs or imprecisions were pointed out by contestants. Starting from February 2010, contestants had access to the evaluation cluster and could check the score of their submission and their ranking. Students had then up to April to work on their implementation. Then, a shortlist of five finalist teams, whose submission had the best performance, was selected and these teams could use the remaining time before the beginning of the conference, in June, to continue improving their implementation.

Setting up the evaluation cluster, eight dedicated PCs running a 64bit version of Linux on a single-core CPU, was not completely straightforward. First, it was critical to ensure the security of the machines and the non-disclosure of the contents of the benchmarks, whereas contestants were allowed to run arbitrary code on the cluster. The solution chosen was to run each submission as a new unprivileged Linux user without any write access to any part of the disk (except for a temporary directory that was emptied after each evaluation), and to set up a strict firewall that prevented any information leaking outside the cluster. Second, for scores to be meaningful, evaluation times had to be reproducible from one run to another. This led us to use dedicated servers rather than virtualization, to clear all system caches across runs, and to ensure, as much as possible, that no concurrent processes were active on the cluster nodes. Third, it turned out that contestants encountered problems (crashes, timeouts) while running their submissions on the evaluation workloads that they were not able to reproduce on their test benchmarks. To help them with debugging, we provided them with stack traces and other similar crash information, from the execution log of the evaluation. This, however, could not be automated, because of potential leaks of detailed information about the content of the benchmarks, and resulted in a time-consuming task for the organizers.

A total of 29 teams took part in the competition, from 23 different institutions in 13 countries. An amazing collective effort was put in this contest, with some teams literally dedicating months of working time to the competition. As a result, leading implementations are impressively sophisticated and efficient.

The five finalist teams and their score, computed at the end of the competition, are listed in Table 1. Note that only the top two teams managed to pass all benchmarks in the allocated time. Both of these implementations are interesting: the winning one, *cardinality*'s, is generally faster, but is actually slower than the second-ranked, namely *dbis*'s, on the last workload, which is also the most difficult one.

More details about the results of the competition can be found at the following URL: `http://dbweb.enst.fr/events/sigmod10contest/results/`.

4. KEY IDEAS BEHIND LEADING IMPLEMENTATIONS

We present in this section the general ideas behind the two systems we implemented. When not stated otherwise, the description applies to both.

Given the task, query planner and executor are two main components to build. Undoubtedly, both of them can make an enormous impact on the overall performance. Here we discuss key decisions on designing and implementing these components in more detail. We refer to a textbook on database management systems such as [2] for precisions on some of the techniques used.

Query Planning

In order to determine the most efficient physical query plan, we implemented cost-based query optimization. Our objective is to minimize total resource consumption rather than response time because of the many concurrent queries running at the same time. The cost model focuses on network transfer as well as sequential and random disk reads. Incorporating CPU cost would be a natural next step, but we believe that its impact on the quality of the selected plan will be marginal given the performance of the main-memory index and the limited class of supported SQL queries. Also, calibrating parameters for complex models would have been more difficult due to restricted access to the evaluation cluster.

Finalist teams employed various plan search strategies. Team *cardinality* enumerates all possible physical query plans and estimates their costs. Because plans are built bottom-up, a dynamic programming technique is applied in order to avoid building and evaluating the same subplan multiple times. During plan enumeration, uninteresting subplans are pruned according to heuristics that favor index-based operators. These heuristics not only reduce the size of search space but also complements the rather simple cost model. Note that time and space consumption of this exhaustive search is bounded because the contest specifies that at most five tables are joined. On the other hand, team *dbis* first performs an exhaustive search for all possible join orders and then refines the

Table 1: Final score, corresponding time, and proportion of the eight workloads processed

	Team	Institution	Country	Score	Time	Workloads
1.	*cardinality*	Stanford University	USA	88	3min 18s	8/8
2.	*dbis*	TU Kaiserslautern	Germany	98	5min 45s	8/8
3.	*spbu*	Saint-Petersburg University	Russia	108	>12min 17s	7/8
4.	*insa*	INSA Lyon	France	119	>22min 20s	7/8
5.	*bugboys*	KAUST	Saudi Arabia	142	>30min 19s	5/8
	Naïve implementation			207	>45min 31s	2/8

plan with optimal join order using heuristics. Once the optimal join order is chosen, the query planner starts with a basic plan using table scans and block nested-loop joins. Subsequently, alternative plans are derived using as many index-based operators as possible. In this step, a block nested-loop join can be replaced by a merge join when two primary key columns are joined. Likewise, if one or both join columns are indexed, a hash join can be used by treating the given index as a hash-based access structure. A similar procedure applies for access operators; a table scan is substituted by an index scan followed by an optional table seek based on the offset retrieved from the index.

Exploiting partitioning information during query planning turns out to be a crucial observation for passing all benchmarks. Clearly, certain queries can be answered by accessing only relevant partitions because all tables are range-partitioned based on the primary key column. For example, primary key joins do not cover all possible partition pairs, but are processed only between partition pairs whose ranges overlap. Similarly, conditions on the primary key column can eliminate out-of-range partitions. Because accessing multiple partitions usually involves network transfer, skipping one partition can decrease the running time significantly. If partitioning information indicates that a condition is unsatisfiable, we generate a query plan with a no-op operator that produces an empty result set.

Statistics of each partition are gathered during the preprocessing step. We obtain partition sizes and estimate the cardinality and average column sizes by sampling a few first pages in each partition. Also, we count the number of distinct values for indexed columns. Other than these basic statistics, we do not maintain detailed information about data distribution and rely on fixed selectivity factors depending on the type of predicates.

Query Execution: Single Node

Tables stored on disk are accessed through memory-mapped I/O. This strategy has two main advantages. First, we can deploy a high performance buffer pool with little implementation effort because the page cache in the Linux kernel provides this function for us. Even though a naïve approach using an input file stream library also utilizes the page cache, we cannot read the page cache directly. Consequently, excessive memory copy between kernel and user spaces cannot be avoided. Second, we can access columns spanning multiple pages efficiently. Unlike most traditional databases, underlying data files are stored in the comma-separated values (CSV) format. Thus, addressing a column would be complicated unless there is a contiguous address space for an entire file. Note that despite these advantages, this strategy would not have been practical on 32-bit architectures where available address space is considerably smaller.

Query execution uses a simple pull-based, Volcano-style, iterator model [1] where each operator implements Open(), GetNext(), and Close() interfaces. A tuple passed by GetNext() is always represented as a vector of pointer-length pairs. Because we avoid memory copy as much as possible, these pointers usually reference either memory-mapped files or communication buffers.

Query Execution: Multiple Nodes

Send and receive operators fulfill the communication among the master and slave nodes. A receive operator generates a request message by serializing the subplan rooted at its child and ships the message to its corresponding send operator. Upon receiving a request message, the send operator constructs the subplan, executes it, and sends its result back as a response message. Each send operator runs on its own worker thread for concurrent execution. Some finalist teams maintained a thread pool with a fixed number of worker threads. However, a naïve approach without a thread pool also performs well because only hundreds of threads are created and destroyed each second in this setting.

The send and receive operators exchange messages over TCP/IP connections, so efficient TCP communication is vital to the performance. The first challenge to minimize TCP overhead is to amortize TCP connection setup and tear-down cost across multiple request-response pairs. Finalist teams addressed this challenge in slightly different ways, but the main idea is to maintain established TCP connections for reuse. Some teams keep a single TCP connection between each pair of nodes,

while the others initiate a new TCP connection whenever there is no reusable connection. In the contest setting, the latter works better because receive operators do not have to wait for the single TCP connection to be available. Another challenge is to reduce per-message TCP overhead in order to increase application throughput. To achieve this goal, we pack each message into fewer TCP segments by setting the `TCP_CORK` option on all TCP connections. The Linux kernel does not transmit partial TCP segments as long as the `TCP_CORK` option is set and a 200-millisecond timer does not expire. As a result, more network bandwidth is utilized by request and response messages rather than TCP/IP headers. These two techniques yield a substantial performance improvement; benchmark 6 runs more than four times faster.

Message compression is another natural way to save network bandwidth. For request messages, we adopted a variable-length integer code that serializes smaller integers into fewer bytes. This simple encoding is quite effective for request messages because serialized query plans contain many small integers such as node ids and column ids. An optimized variable-length encoding implementation decreases the running times of benchmarks 6 and 7 by more than 20%. For response messages, we experimented with a couple of lossless compression algorithms. Even though we observed good compression ratios and some performance improvements using our test dataset and queries, we failed to decrease any benchmark running time. This result is not surprising because the cost of message compression can be more expensive than the benefit depending on workload and hardware configuration. Clearly, we need a more sophisticated algorithm that determines which messages to compress in order for this technique to work well across various workloads.

5. CLOSING REMARKS

The SIGMOD 2009 and 2010 programming contests were a chance for many students (at both the undergraduate and graduate levels) to discover and design parts of the architecture of a distributed database management system. The contest was used in several universities as part of the curriculum or as an optional alternative to other assignments. This programming contest will run again next year, organized by Stavros Harizopoulos and Mehul Shah from HP Labs. It is our hope that this competition will help foster the next generation of database researchers and practitioners.

6. ACKNOWLEDGMENTS

We are very grateful to the sponsors of the programming contest: NSF, Microsoft (platinum sponsors); Amazon, INRIA Saclay (gold sponsors); Exalead, Yahoo! (silver sponsors). We would also like to acknowledge our advisory board: Serge Abiteboul, Magdalena Balazinska, Samuel Madden, and Michael Stonebraker.

7. REFERENCES

[1] Goetz Graefe. Volcano - an extensible and parallel query evaluation system. *IEEE Trans. Knowl. Data Eng.*, 6(1):120–135, 1994.

[2] Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems*. McGraw-Hill, New York, USA, third edition, 2002.