SIGMOD Officers, Committees, and Awardees

Chair

Yannis Ioannidis
University of Athens
Department of Informatics
Panepistimioupolis, Informatics Bldg
157 84 Ilissia, Athens
HELLAS
+30 210 727 5224
<yannis AT di.uoa.gr>

Vice-Chair

Christian S. Jensen
Department of Computer Science
Aarhus University
Åbogade 34
DK-8200 Århus N
DENMARK
+45 99 40 89 00
<csj AT cs.aau.dk >

Secretary/Treasurer

Alexandros Labrinidis
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260-9161
PA 15260-9161
USA
+1 412 624 8843
<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>,
Amr El Abbadi, Serge Abiteboul, Rakesh Agrawal, Anastasia Ailamaki, Ricardo Baeza-Yates, Phil Bernstein, Elisa Bertino, Mike Carey, Surajit Chaudhuri, Christos Faloutsos, Alon Halevy, Joe Hellerstein, Masaru Kitsuregawa, Donald Kossmann, Renée Miller, C. Mohan, Beng-Chin Ooi, Meral Ozsoyoglu, Sunita Sarawagi, Min Wang, and Gerhard Weikum.

Information Director, SIGMOD DiSC and SIGMOD Anthology Editor:

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

Associate Information Directors:

Ugur Cetintemel, Manfred Jeusfeld, Georgia Koutrika, Alexandros Labrinidis,
Michael Ley, Wim Martens, Rachel Pottinger, Altigran Soares da Silva, and Jun Yang.

SIGMOD Record Editor-in-Chief:

Ioana Manolescu, Inria Saclay—Île-de-France, <ioana.manolescu AT inria.fr>

SIGMOD Record Associate Editors:

Yanif Ahmad, Denilson Barbosa, Pablo Barceló, Vanessa Braganholo, Marco Brambilla, Chee Yong Chan, Anish Das Sarma, Glenn Paulley, Alkis Simitsis, Nesime Tatbul and Marianne Winslett.

SIGMOD Conference Coordinator:

Sihem Amer-Yahia, CNRS and LIG, France, <sihemameryahia AT acm.org>

PODS Executive Committee: Rick Hull (chair), <hull AT research.ibm.com>, Michael Benedikt, Wenfei Fan, Maurizio Lenzerini, Jan Paradaens and Thomas Schwentick.

Sister Society Liaisons:

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

Awards Committee:

Rakesh Agrawal (Chair), Microsoft Research, <rakesh.agrawal AT microsoft.com>, Elisa Bertino, Peter Buneman, Umesh Dayal 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 : September 15th, 2012]

Alfons Kemper, Hank Korth, Alberto Laender, Boon Thau Loo, Timos Sellis, and Kyu-Young Whang.

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)	Surajit Chaudhuri (2011)	Bruce Lindsay (2012)

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)	Gerhard Weikum (2011)	Marianne Winslett (2012)

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.* Recipients of the award are the following:

- **2006** *Winner*: Gerome Miklau, University of Washington. *Runners-up*: Marcelo Arenas, University of Toronto; Yanlei Diao, University of California at Berkeley.
- **2007** *Winner*: Boon Thau Loo, University of California at Berkeley. *Honorable Mentions*: Xifeng Yan, University of Indiana at Urbana Champaign; 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, University of Washington. *Honorable Mentions*: Soumyadeb Mitra, University of Illinois, Urbana-Champaign; Fabian Suchanek, Max-Planck Institute for Informatics.
- **2011** *Winner*: Stratos Idreos, Centrum Wiskunde & Informatica. *Honorable Mentions*: Todd Green, University of Pennsylvania; Karl Schnaitter, University of California in Santa Cruz.
- **2012** *Winner*: Ryan Johnson, Carnegie Mellon University. *Honorable Mention*: Bogdan Alexe, University of California in Santa Cruz.

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

[Last updated : June 30th, 2012]

Editor's Notes

Welcome to the September 2012 issue of the ACM SIGMOD Record!

The issue opens with the Database Principles column, where Karvounarakis and Green survey the recent literature on querying semiring-annotated data. Annotated data models have been proposed for various applications, including provenance, probabilistic and incomplete information. An earlier work (PODS 2007) by Karvounarakis, Green and Tannen had proposed the general framework of *K*-relations, which captures the querying of such annotated relations, where *K* is a commutative semiring. The present paper surveys the literature on querying semiring-annotated data and presents new development of the ideas, based on semiring of polynomials, applying in particular to provenance representations and to XQuery views over annotated, unordered XML.

The survey by Manica, Dorneles and Galante focuses on the support for the time dimension within Web search engines. Users may either issue search queries with explicit time constraints, or query data related to the "recent past" (what is the latest version of this information?") or forecast data (what are the predictions for tomorrow's weather?) Successive generations of Web search engines have tackled the time dimension from different angles, from providing the freshest results first, to allowing the user to specify a time window for their results, and finally to extract and manipulate the temporal information comprised in Web pages. The authors end with a set of currently open problems and perspectives on further features, and more efficiency, that could be achieved when answering search queries with a temporal dimension.

Ryan Johnson is the recipient of the SIGMOD 2012 Doctoral Dissertation award (advised by Anastasia Ailamaki). In the Distinguished Database Profiles column, he talks about the importance of understanding experimental results, of the secrets and research avenues that lie hidden just behind an unexpected shape of a curve, and of the benefits of crossing out of one's immediate thematic community to learn from others and possibly understand how their results affect our problems. One could moreover congratulate Ryan for a very relaxed, yet probably very wise, approach for his first job hunt; read the column to find out!

In the Research Centers column, Palpanas and Velegrakis describe their successful database group in Trento (dbTrento). The group is currently 17-strong, and is working on topics at the intersection of database and semantics: semantic-based keyword search, approximate query answering, entity resolution, schema and data mapping, benchmarking and updates within Information Integration systems. Importantly, this Trento group will host next year's VLDB conference – an extra good reason to get to know more about the group through this issue's column.

The Industrial Perspective column features an article by Kulkarni and Michels on the temporal features in the new SQL:2011 standard. The fundamental feature introduced in the language is the association of a period definition attached as metadata to tables. To correctly handle multiple temporal versions of an information item, information about the application-time period must also be included in declared primary keys. The paper further discusses referential integrity constraints and querying for tables with application-time periods and for system-versioned tables, and briefly compares the standard to previous temporal models. The mapping is not straightforward, thus this short paper is very welcome for those familiar with temporal databases concepts and seeking to use the new ISO standard.

Two event reports appear in this issue. First, Kissinger, Schlegel, Boehm, Habisch and Lehner provide insights into their winning solution of the ACM SIGMOD Programming Contest 2011. The challenge was to build a high-throughput in-memory index, durable on Flash-based storage. Their approach combined high-throughput for this order-preserving in-memory structure with an optimized logging mechanism to

support the hard durability requirements of the contest. I commend this short report for the very clear writing style, detailed at just the right level, rich with examples and a compelling motivation for the problem being targeted. Second, Mazón, Garrigós, Daniel and Castellanos report on the BEWEB (Business Intelligence and the Web) workshop held next to EDBT 2011. The workshop considered technical aspects of today's BI, including handling (highly) heterogeneous data, running BI analysis over Web data, and engineering Web-based BI applications.

The issue closes with the general call for contributions to the ACM SIGMOD 2013 conference, the more specific call for Research Papers submission to SIGMOD 2013, introducing a journal-style two-cycle review process, and the PODS 2013 call for papers. Check out the details and prepare for New York!

Changes to the editorial board The Record has continued the renewal of its editorial board. After several years of outstanding service editing our columns, four editors have stepped out, after passing over the tricks of the trade to new recruits replacing them:

- Magdalena Balazinska leaves the Systems and Prototypes column to Marco Brambilla, assistant professor at Politecnico di Milano, Italy.
- Ugur Cetintemel trusts Alkis Simitsis to handle the Research Centers column all by himself now, the trust is well placed given the impeccable job Alkis did on this issue's column!
- Brian Cooper has passed his skills for handling Event Report submissions to Anish Das Sarma.
- Cesar Galindo-Legaria has handed the Survey column responsibility to Nesime alone.

Last but not least, to help us process regular research articles, Yanif Ahmad, assistant professor at Johns Hopkins University in Baltimore, USA, also joined the editorial board.
Many thanks go to the three outgoing editors for their contributions to the Record, and best wishes for a long and satisfying tenure to the new editors!

Your contributions to the Record are welcome via the RECESS submission site (http://db.cs.pitt.edu/recess). Prior to submitting, be sure to peruse the Editorial Policy on the SIGMOD Record's Web site (http://www.sigmod.org/publications/sigmod-record/sigmod-record-editorial-policy).

Ioana Manolescu

September 2012

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)

Semiring-Annotated Data: Queries and Provenance*

Grigoris Karvounarakis
LogicBlox, Inc
1349 W Peachtree St NW
Atlanta, GA 30309
grigoris.karvounarakis@logicblox.com

Todd J. Green
LogicBlox, Inc
1349 W Peachtree St NW
Atlanta, GA 30309
todd.green@logicblox.com

ABSTRACT

We present an overview of the literature on *querying semiring-annotated data*, a notion we introduced five years ago in a paper with Val Tannen. First, we show that positive relational algebra calculations for various forms of annotated relations, as well as provenance models for such queries, are particular cases of the same general algorithm involving commutative semirings. For this reason, we present a formal framework for answering queries on data with annotations from commutative semirings, and propose a comprehensive provenance representation based on semirings of *polynomials*. We extend these considerations to XQuery views over annotated, unordered XML data, and show that the semiring framework suffices for a large positive fragment of XQuery applied to such data. Finally, we conclude with a brief overview of the large body of work that builds upon these results, including both extensions to the theoretical foundations and uses in practical applications.

1. INTRODUCTION

Various forms of *annotated data models* have appeared over the years in the database theory literature, ranging from the study of incomplete [30, 20, 21] and probabilistic [13, 33] databases, to query answering under bag (multiset) semantics and to various models to record the provenance of query results in data sharing and warehousing settings [10, 6, 9, 18, 8]. In a 2007 paper with Val Tannen [19], we showed that many of the mechanisms for evaluating queries over such annotated relations can be unified in a general framework based on *K-relations*, which are relations whose tuples are annotated with elements from a commutative semiring K.

The semantics of positive relational algebra ($\mathcal{RA}^+$) queries extends to K-relations via definitions in terms of the abstract "+" and "·" operations of K. Intuitively, "+" corresponds to alternative use of data in producing a query result, while "·" corresponds to joint use. For $K = \mathbb{B}$, the Boolean semiring, this specializes to the usual set semantics, while for $K = \mathbb{N}$, the semiring of natural numbers, it is bag semantics.

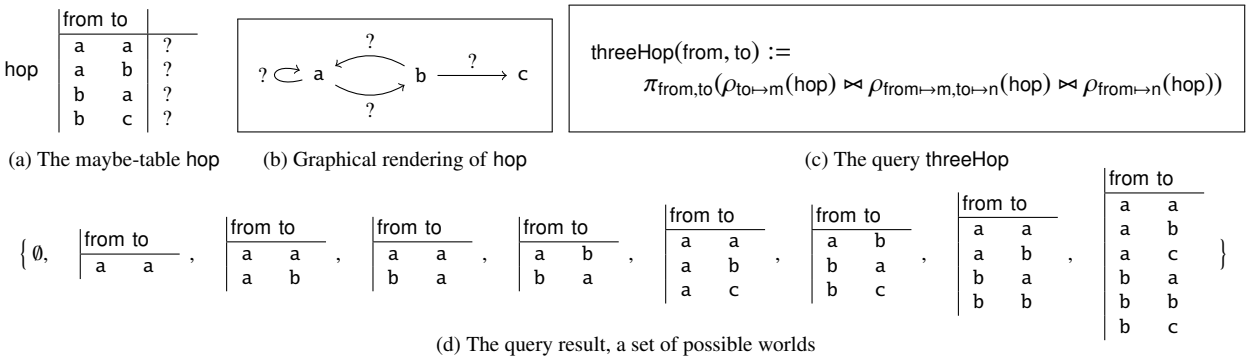
In fact, it turns out that the laws of commutative semirings are *forced* by certain expected identities of $\mathcal{RA}^+$. As additional evidence for the robustness of semiring annotations, we showed with Tannen [19] that the framework extends naturally to recursive Datalog queries (for semirings in which infinite sums are well-defined), and with Nate Foster [12], we gave an extension to a large fragment of positive XQuery over annotated unordered XML data. In each case, we proved a *fundamental theorem* stating that the semantics of queries commutes with the application of semiring homomorphisms. This theorem provides the formal backbone for applications in incomplete and probabilistic databases, trust evaluation, security applications, and others.

A corollary of the fundamental theorem is that, in all these cases, we can use a symbolic *provenance polynomial* representation of semiring calculations to record, document and track the provenance (lineage) of query answers. Using such "most general" semirings as provenance models has the benefit that annotation computations on various semirings factor through them. The resulting provenance expressions can be used at any time to compute result annotations from various semirings as well as for different assignments of values from those semirings to source data (e.g., corresponding to different beliefs of various users about the trustworthiness or quality of source data).

Since the publication of these two papers on semiring-annotated relations and XML, a number of followup studies have investigated semantics and provenance models for additional query operators [14, 3, 4, 17] and data models [31], the interaction of provenance information with query optimization [15, 26], minimization [2] and factorization [29] of provenance expressions, and other topics. Moreover, K-relations have been incorporated

*Portions of this column were adapted from joint work with Val Tannen [19] and Nate Foster and Val Tannen [12].

Database Principles Column. Column editor: Pablo Barceló, Department of Computer Science, University of Chile. E-mail: pbarcelo@dcc.uchile.cl.



(a) The maybe-table hop (b) Graphical rendering of hop (c) The query threeHop

(d) The query result, a set of possible worlds

Figure 1: Maybe-tables and query result. Table hop represents possible links between nodes in a network. The query threeHop computes pairs (x, y) of nodes such that y is reachable from x in exactly three hops. Operator ρ is the rename operator, used in conjunction with the natural join operator $\bowtie$ to express equijoins.

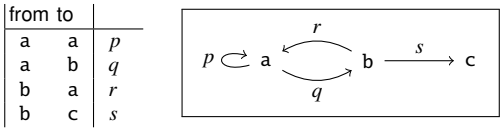
into research systems prototypes to support features including provenance-based trust evaluation [18, 25] and other forms of provenance querying [25], view update [24], maintenance [18, 16] and adaptation strategies [16]. Finally, they have been incorporated into the LogicBlox engine [28] (a commercial declarative programming platform supporting applications in retail planning and analytics), where provenance is used to aid Datalog program analysis and debugging.

The rest of this column is organized as follows. In Section 2, we reprise some original motivations of our work with Tannen [19] by illustrating the similarities of query answering on annotated relations through an example. Next, we define the semantics of $\mathcal{RA}^+$ queries on relations annotated with elements from a semiring K (Section 3), and propose *polynomials*, the *most general* commutative semiring, as a suitable provenance model for $\mathcal{RA}^+$ queries (Section 4). We sketch the extension to annotated XML data [12] in Section 5. Finally (Section 6), we discuss further extensions to incorporate negation, aggregation, and other considerations.

2. ANNOTATED RELATIONS

We motivate our study by considering three important examples of query answering on annotated relations and highlight the similarities between them.

The first example comes from the study of *incomplete databases*, where a simple representation system is the *maybe-table* [30, 20], in which optional tuples are annotated with a '?'. Such a table represents a set of *possible worlds*, capturing the fact that some of the information in the database may be missing or incorrect. Table hop in Figure 1a is a maybe-table representing possible links between nodes in a network (rendered graphically in Figure 1b). The answer to a query over such tables is the set of instances obtained by evaluating the query over each possible world. For example, the result of the query threeHop in Figure 1c is the set of possible worlds shown in Figure 1d. Unfortunately, this set of possible



(a) C-table hop and its graphical rendering

from to	
a a	$(p \wedge p \wedge p) \vee (p \wedge q \wedge r) \vee (p \wedge q \wedge r) \equiv p$
a b	$(p \wedge p \wedge q) \vee (q \wedge q \wedge r) \equiv (p \wedge q) \vee (q \wedge r)$
a c	$p \wedge q \wedge s$
b a	$(p \wedge r \wedge r) \vee (q \wedge r \wedge r) \equiv (p \wedge r) \vee (q \wedge r)$
b b	$p \wedge q \wedge r$
b c	$q \wedge r \wedge s$

(b) C-table query result threeHop

Figure 2: Example of data annotated with Boolean variables, and result of Imielinski-Lipski computation.

worlds cannot itself be represented by a maybe-table. For instance, observe that whenever the tuple (a, c) appears in the result, so does (a, b) and (a, a), and maybe-tables cannot represent such a dependency.

To overcome such limitations, Imielinski and Lipski [21] introduced *c-tables*, where tuples are annotated with Boolean formulas called *conditions*. A maybe-table is a simple kind of c-table, where the annotations are distinct Boolean variables, as shown in Figure 2a. In contrast to weaker representation systems, c-tables are expressive enough to be *closed* under $\mathcal{RA}$ queries, and the main result of Imielinski and Lipski [21] is an algorithm for answering $\mathcal{RA}$ queries on c-tables, producing another c-table as a result. On our example, this algorithm produces the c-table threeHop in Figure 2b; this c-table represents exactly the set of possible worlds shown in Figure 1d. As shown in Figure 2b, some of the Boolean formulas in this c-table can be simplified by applying standard Boolean algebra identities.

Another kind of table with annotations is a *multiset* or *bag*. In this case, the annotations are natural numbers which represent the multiplicity of the tuple in the multiset. (A tuple not listed in the table has multiplicity 0.)

from	to	
a	a	1
a	b	4
b	a	2
b	c	3

(a) Bag table hop

from	to	
a	a	$1\cdot1\cdot1+1\cdot4\cdot2+1\cdot4\cdot2=17$
a	b	$1\cdot1\cdot4+4\cdot4\cdot2=36$
a	c	$1\cdot4\cdot3=12$
b	a	$1\cdot1\cdot2+4\cdot2\cdot2=18$
b	b	$1\cdot4\cdot2=8$
b	c	$4\cdot2\cdot3=24$

(b) Bag query result threeHop

Figure 3: Bag semantics example

from	to	
a	a	x
a	b	y
b	a	z
b	c	u

E	Pr
x	0.6
y	0.5
z	0.1
u	0.4

(a) Probabilistic event table hop

from	to	
a	a	x
a	b	$(x\cap y)\cup(y\cap z)$
a	c	$x\cap y\cap u$
b	a	$(x\cap z)\cup(y\cap z)$
b	b	$x\cap y\cap z$
b	c	$y\cap z\cap u$

(b) Event table threeHop

Figure 4: Probabilistic example

Query answering on such tables involves calculating not just tuples in the output, but also their multiplicities.

For example, consider the multiset shown in Figure 3a. The result of the query threeHop from Figure 1c is the multiset shown in Figure 3b. Note that for projection and union we add multiplicities, while for join we multiply them. There is a striking similarity between the arithmetic calculations we do here for multisets, and the Boolean calculations for the c-table (before the simplifications due to Boolean algebra identities).

A third example comes from the study of *probabilistic databases*, where tuples are associated with values from $[0, 1]$ which represent the probability that the tuple is present in the database. Answering queries over probabilistic tables requires computing the correct probabilities for tuples in the output. To do this, Fuhr and Röllecke [13] and Zimányi [33] (FRZ) introduced *event tables*, where tuples are annotated with probabilistic events, and they gave a query answering algorithm for computing the events associated with tuples in the query output.

Figure 4a shows an example of an event table, with associated *event probabilities* (e.g., z represents the event that the link (b, a) exists). Considering again the same query threeHop as above, the FRZ query answering algorithm produces the event table shown in Figure 4b. Note again the similarity between this table and the example earlier with c-tables. The probabilities of tuples in the output of the query can be computed from this table using the independence of various events that appear together in each expression.

3. POSITIVE RELATIONAL ALGEBRA

In this section we attempt to unify the examples above by considering generalized relations in which the tuples are annotated (*tagged*) with information of various kinds. Then, we will define a generalization of the positive relational algebra ($\mathcal{RA}^+$) to such tagged-tuple relations. The examples in Section 2 will turn out to be particular cases.

We use here the named perspective [1] of the relational model in which tuples are functions $t : U \to \mathbb{D}$ with U a finite set of attributes and $\mathbb{D}$ a domain of values. We fix the domain $\mathbb{D}$ for the time being and we denote the set of all such U-tuples by U-Tup. (Usual) relations over U are subsets of U-Tup.

A notationally convenient way of working with tagged-tuple relations is to model tagging by a function on all possible tuples, with those tuples not considered to be "in" the relation tagged with a special value. For example, the usual set-theoretic relations correspond to functions that map U-Tup to $\mathbb{B} = \{\mathsf{true}, \mathsf{false}\}$ with the tuples in the relation tagged by true and those not in the relation tagged by false.

Definition 3.1. Let K be a set containing a distinguished element 0. A **K-relation** over a finite set of attributes U is a function $R : U\text{-Tup} \to K$ such that its **support** defined by $\mathsf{supp}(R) \stackrel{\text{def}}{=} \{t \mid R(t) \neq 0\}$ is finite.

In generalizing $\mathcal{RA}^+$ we will need to assume more structure on the set of tags. To deal with selection we assume that the set K contains two distinct values $0 \neq 1$ which denote "out of" and "in" the relation, respectively. To deal with union and projection and therefore to combine different tags of the same tuple into one tag we assume that K is equipped with a binary operation "+". To deal with natural join (hence intersection and selection) and therefore to combine the tags of joinable tuples we assume that K is equipped with another binary operation "$\cdot$".

Definition 3.2. Let $(K, +, \cdot, 0, 1)$ be an algebraic structure with two binary operations and two distinguished elements. The main operations of the **positive algebra** are defined as follows:

union If $R_1, R_2 : U\text{-Tup} \to K$ then $R_1 \cup R_2 : U\text{-Tup} \to K$ is defined by: $(R_1 \cup R_2)(t) \stackrel{\text{def}}{=} R_1(t) + R_2(t)$

projection If $R : U\text{-Tup} \to K$ and $V \subseteq U$ then $\pi_V R : V\text{-Tup} \to K$ is defined by:

$$(\pi_V R)(t) \stackrel{\text{def}}{=} \sum_{t=t' \text{ on } V \text{ and } R(t')\neq 0} R(t')$$

natural join If $R_i : U_i\text{-Tup} \to K$, $(i = 1, 2)$, then $R_1 \bowtie R_2$ is the K-relation over $U_1 \cup U_2$ defined by:

$$(R_1 \bowtie R_2)(t) \stackrel{\text{def}}{=} R_1(t_1) \cdot R_2(t_2)$$

where $t_1 = t$ on U_1 and $t_2 = t$ on U_2

In our paper with Tannen [19] we also provided definitions for the remaining operators, such as renaming ρ and selection σ, and showed that the resulting definition gives us an algebra on K-relations and generalizes the definitions of $\mathcal{RA}^+$ for the examples in Section 2.

Indeed, for $(\mathbb{B}, \vee, \wedge, \mathsf{false}, \mathsf{true})$ we obtain the usual $\mathcal{RA}^+$ with set semantics. For $(\mathbb{N}, +, \cdot, 0, 1)$ it is $\mathcal{RA}^+$ with bag semantics. For the Imielinski-Lipski algebra on c-tables we consider the semiring $(\mathsf{PosBool}(B), \vee, \wedge, \mathsf{false}, \mathsf{true})$ of *positive* Boolean expressions over some set B of variables, i.e., $\neg$ is disallowed. (We identify those expressions that yield the same truth-value for all Boolean assignments of the variables in B.) Applying Definition 3.2 to the structure produces exactly the Imielinski-Lipski algebra. Finally, for $(\mathcal{P}(\Omega), \cup, \cap, \emptyset, \Omega)$ we obtain the FRZ $\mathcal{RA}^+$ on event tables.

These four structures are examples of *commutative semirings*, i.e., algebraic structures $(K, +, \cdot, 0, 1)$ such that $(K, +, 0)$ and $(K, \cdot, 1)$ are commutative monoids, $\cdot$ is distributive over $+$, and for all a, we have $0 \cdot a = a \cdot 0 = 0$.

Other examples of commutative semirings include:

- $(\mathbb{N}^\infty, \min, +, \infty, 0)$, the *tropical semiring* [27], where we add ∞ to the natural numbers. In our running example, we might annotate the edges of hop with elements of $\mathbb{N}^\infty$ representing costs; then, a tuple (x, y) in threeHop would be annotated by the cost of the cheapest path of length three from x to y.
- $([0, 1], \max, \min, 0, 1)$ which is related to *fuzzy sets* [32], and could be called the *fuzzy semiring*.
- The semiring of *confidentiality policies* [12] $(C, \min, \max, \mathsf{P}, 0)$, where the total order $C = \mathsf{P} < \mathsf{C} < \mathsf{S} < \mathsf{T} < 0$ describes levels of security clearance: P public, C confidential, S secret, and T top-secret.

Further evidence for requiring K to form such a semiring is given by a result in our paper with Tannen [19] stating that certain standard $\mathcal{RA}^+$ identities (such as commutativity of joins and unions) hold for the positive algebra on K-relations iff $(K, +, \cdot, 0, 1)$ is a commutative semiring. The list of relational identities does not include the idempotence of unions and joins, since these fail for bag semantics, an important special case in our treatment.

Any function $h\colon K \to K'$ can be used to transform K-relations to K'-relations simply by applying h to each tag (note that the support may shrink but never increase). Abusing the notation a bit we denote the resulting transformation from K-relations to K'-relations also by h. We can now state the following *fundamental theorem*, showing that the $\mathcal{RA}^+$ operations work nicely with semiring structures.

Theorem 3.3 (Fundamental Theorem)**.** *Let $h\colon K \to K'$ and assume that K, K' are commutative semirings. The transformation given by h from K-relations to K'-relations commutes with any $\mathcal{RA}^+$ query (i.e., $q(h(R)) = h(q(R))$) iff h is a semiring homomorphism.*

Example 3.4 (Incomplete databases). The c-table hop of Figure 2a represents the set of possible worlds

$$\{h_\nu(\mathsf{hop}) \mid \nu : B \to \mathbb{B}\}$$

where $h_\nu : \mathsf{PosBool}B \to \mathbb{B}$ is the semiring homomorphism which "plugs in" values for the Boolean variables according to ν then simplifies the result to true or false. Theorem 3.3 tells us that the c-table threeHop of Figure 2b correctly represents the possible worlds of Figure 1d. □

Example 3.5 (Non-interference). Recall the semiring C of confidentiality policies. For a user with a given security clearance level c, we want to present a view of the database containing only tuples of clearance level $c' \leq c$. "Erasing" tuples exceeding a given clearance level c corresponds to applying the semiring homomorphism $h_c : C \to C$ which maps $c' \leq c$ to c' and to 0 otherwise. Theorem 3.3 implies that we get the same result by evaluating the query first, then erasing unauthorized tuples from the result, that we get by erasing first, then evaluating the query. □

4. POLYNOMIALS FOR PROVENANCE

Apart from the kinds of annotations we have discussed until now, an important category involves *provenance models* [8], which have been defined as a way of relating the tuples in a query output to the tuples in the input that "contribute" to them. For this reason, in our paper with Tannen [19] we proposed using the different operations of the semiring from Definition 3.2, which appear to fully "document" how an output tuple is produced. To record the documentation as tuple tags we need to use a semiring of symbolic expressions. In the case of semirings, like in ring theory, these are the *polynomials*.

Definition 4.1. Let X be the set of tuple ids of a (usual) database instance I. The *positive algebra provenance semiring* for I is the semiring of polynomials with variables (a.k.a. indeterminates) from X and coefficients from $\mathbb{N}$, with the operations defined as usual[1]:

$$(\mathbb{N}[X], +, \cdot, 0, 1)$$

Example 4.2. Start again from our running example, with tuples tagged with their own id. These relations can be seen as $\mathbb{N}[p, q, r, s]$-relations. Applying the query threeHop from Section 2 and performing the calculations in the provenance semiring we obtain the annotations in the last column of Figure 5. The provenance of (a, a) is $p^3 + 2pqr$ which can be "read" as follows: (a, a) is derived by threeHop in three different ways; one of them uses the input tuple annotated with p *three* times; the other two both involve joining input tuples annotated with p, q and r. □

The following property of polynomials captures the intuition that $\mathbb{N}[X]$ is as "general" as any semiring.

[1]These are polynomials in commutative variables so their operations are the same as in middle-school algebra, except that subtraction is not allowed.

from	to	Lin(X)	Why(X)	Trio(X)	$\mathbb{B}[X]$	$\mathbb{N}[X]$
a	a	pqr	$p + pqr$	$p + 2pqr$	$p^3 + pqr$	$p^3 + 2pqr$
a	b	pqr	$pq + qr$	$pq + qr$	$p^2q + q^2r$	$p^2q + q^2r$
a	c	pqs	pqs	pqs	pqs	pqs
b	a	pqr	$pr + qr$	$pr + qr$	$p^2r + qr^2$	$p^2r + qr^2$
b	b	pqr	pqr	pqr	pqr	pqr
b	c	qrs	qrs	qrs	qrs	qrs

Figure 5: Five kinds of provenance for threeHop

Proposition 4.3. *Let K be a commutative semiring and X a set of variables. For any valuation $v : X \to K$ there exists a unique homomorphism of semirings*

$$\mathsf{Eval}_v : \mathbb{N}[X] \to K$$

such that for one-variable monomials $\mathsf{Eval}_v(x) = v(x)$.

$\mathsf{Eval}_v(P)$ evaluates the polynomial P in K given a valuation for its variables. In calculations with integer coefficients, na where $n \in \mathbb{N}$ and $a \in K$ is the sum in K of n copies of a. Note that $\mathbb{N}$ is embedded in K by mapping n to the sum of n copies of 1_K.

Using the Eval notation, for any $P \in \mathbb{N}[x_1, \ldots, x_n]$ and any K the *polynomial function* $f_P : K^n \to K$ is given by:

$$f_P(a_1, \ldots, a_n) \stackrel{\text{def}}{=} \mathsf{Eval}_v(P) \quad v(x_i) = a_i,\ i = 1..n$$

Putting together Propositions 3.3 and 4.3 we obtain Theorem 4.4 below, a conceptually important fact that says, informally, that the semantics of $\mathcal{RA}^+$ on K-relations for any semiring K *factors* through the semantics of the same in provenance semirings.

Indeed, let K be a commutative semiring, let R be a K-relation, and let X be the set of tuple ids of the tuples in $\mathsf{supp}(R)$. There is an obvious valuation $v : X \to K$ that associates to a tuple id the tag of that tuple in R.

We associate to R an "abstractly tagged" version, denoted $\bar{R}$, which is an $X \cup \{0\}$-relation. $\bar{R}$ is such that $\mathsf{supp}(\bar{R}) = \mathsf{supp}(R)$ and the tuples in $\mathsf{supp}(\bar{R})$ are tagged by their own tuple id. Note that as an $X \cup \{0\}$-relation, $\bar{R}$ is a particular kind of $\mathbb{N}[X]$-relation.

To simplify notation we state the following corollary of Theorem 3.3 for queries of one argument (but the generalization is immediate):

Corollary 4.4. *For any $\mathcal{RA}^+$ query q we have*

$$q(R) \;=\; \mathsf{Eval}_v \circ q(\bar{R})$$

To illustrate an instance of this theorem, consider the provenance polynomial $p^3 + 2pqr$ of the tuple (a, a) in the last column of the table in Figure 5. Evaluating it in $\mathbb{N}$ for $p = 1, q = 4, r = 2, s = 3$ we get 17 which is indeed the multiplicity of (a, a) in Figure 3.

4.1 Provenance hierarchy

Apart from the various forms of annotations described earlier, it turns out that various provenance models can also be captured by semirings, and thus provenance polynomials generalize those models as well. In fact, these

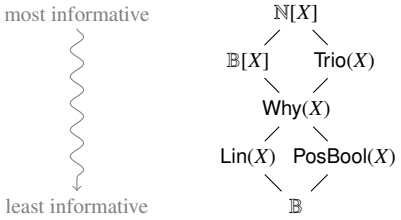


Figure 6: Provenance hierarchy. A path downward from K_1 to K_2 indicates that there exists a surjective semiring homomorphism from K_1 to K_2.

models of provenance can be neatly arranged in the hierarchy of Figure 6 [15].

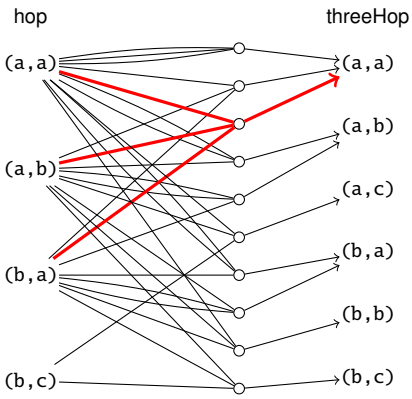
In this figure, $\mathbb{B}[X]$ denotes the *Boolean provenance polynomials semiring* [15] over variables X, Trio(X) denotes a semiring capturing the form of provenance used in the *Trio* system [15, 5], and Why(X) corresponds to *why-provenance* [6]. Formal definitions can be found in the paper by Green [15], but intuitively, annotations in $\mathbb{B}[X]$ (Trio(X), respectively) can be obtained from the provenance polynomials by dropping coefficients (exponents, resp.); while annotations from Why(X) can be obtained by dropping both coefficients and exponents. Annotations from the semiring Lin(X) capturing *lineage* [10] collapse all variables appearing in the polynomial into a single monomial. The corresponding provenance expressions for these models in our running example are shown in Figure 5.

Coupled with Proposition 3.3, this hierarchy gives a clear picture of the relative "*informativeness*" of the various provenance models, since provenance computations for models lower in the hierarchy can always be factored through computations involving models above them in the hierarchy. This additional informativeness of provenance polynomials, compared to all other provenance models in the literature, allows to overcome some of their limitations [19] (also recognized in SPIDER [9] for lineage) to support a variety of applications on annotated data.

4.2 Provenance as a graph

We conclude our discussion of annotated relations by presenting an alternative *graphical* interpretation of provenance polynomials. This viewpoint is especially useful in practical systems needing provenance support such as Orchestra [18] and LogicBlox [28].

We define a *provenance graph* for a query as having two kinds of nodes, *tuple nodes*, one for each source or derived tuple, and *join nodes*. Edges connect tuple nodes to join nodes. Intuitively, each join node corresponds to the instantiation of an n-way join in the query, with incoming edges from tuple nodes participating in the join, and a single outgoing edge to the tuple node output by the join. Output tuple nodes may have multi-



(a)

prThreeHop

from	m	n	to
a	a	a	a
a	a	b	a
a	b	a	a
a	a	a	b
a	b	a	b
a	a	b	c
b	a	a	a
b	a	b	a
b	a	a	b
b	a	b	c

(b)

```
threeHop(X,Y) :- hop(X,M), hop(M,N), hop(N,Y).
```

(c)

```
prThreeHop(X,M,N,Y) :- hop(X,M), hop(M,N), hop(N,Y).
threeHop(X,Y) :- prThreeHop(X,_,_,Y).
```

(d)

Figure 7: Graphical representation of provenance polynomials, along with the graph's relational encoding, and the Datalog rules which construct the graph.

ple incoming edges, representing alternative derivations.

For example, Figure 7a shows the graphical representation of the provenance polynomials in Figure 5. The left column contains tuple nodes for hop, the right column has tuple nodes for threeHop, and the center column has connecting join nodes. The highlighted edges in the graph correspond to (one copy of) the monomial pqr in the annotation $p^3 + 2pqr$ for output tuple (a, a).

This graph representation has served as the basis for a compact storage scheme for provenance. Indeed, with some extensions described in Section 6, this graph model has been used for provenance storage and querying [18, 25] in the ORCHESTRA collaborative data sharing system (CDSS). In ORCHESTRA data is propagated from various sources through paths of *schema mappings* (akin to Datalog rules) and materialized in data warehouses, and provenance is used for view maintenance [18] and update [24], as well as trust evaluation [18, 25]. A similar provenance storage scheme, as well as provenance querying functionality, is supported by the LogicBlox [28] Datalog engine. Figure 8 illustrates the output of a couple of provenance queries issued through the command-line LogicBlox client for a Datalog program encoding our running example.

We illustrate how these systems store provenance graphs in relations on our running example. A query such as threeHop (shown in Figure 7c in Datalog syntax) is translated into a pair of Datalog rules shown in Figure 7d. From the first rule, the relation prThreeHop contains one tuple for every join node in the provenance graph. The resulting provenance relation for the graph of Figure 7a is shown in Figure 7b (where we follow the

```
$ bloxbatch -interactive
<blox>: provenance <doc>
          +provenance[`threeHop]("a","a").
        </doc>
Provenance information:
threeHop(a,a) <- hop(a,a),hop(a,a),hop(a,a).
threeHop(a,a) <- hop(a,a),hop(a,b),hop(b,a).
threeHop(a,a) <- hop(a,b),hop(b,a),hop(a,a).
<blox>: provenance <doc>
          +provenance[`threeHop]("a","b").
        </doc>
Provenance information:
threeHop(a,b) <- hop(a,a),hop(a,a),hop(a,b).
threeHop(a,b) <- hop(a,b),hop(b,a),hop(a,b).
<blox>:
```

Figure 8: Provenance queries in LogicBlox command-line client, for a Datalog program encoding our running example.

order of the join nodes in Figure 7a, and we use horizontal lines to distinguish derivations producing the same result tuple, e.g., the first three tuples correspond to the top three join nodes, that encode derivations for the tuple (a,a)). The highlighted tuple in the table encodes the highlighted edges in the graph.

The storage scheme described above avoids storing redundantly some common provenance subexpressions [25, 23], which are ubiquitous in settings with multiple queries, such as large Datalog programs or complex data sharing settings. Finally, as we explain in Section 6, the graph—as well as its relational encoding—provides a finite representation for the (possibly infinite) provenance of the results of recursive Datalog queries. Even though the graph representation does not store provenance polynomials directly, one can generate them by traversing the graph recursively backwards along its edges (this can be achieved by joining provenance relations) [25].

5. ANNOTATED XML

In this section, we sketch a generalization of K-relations beyond flat relational data by Foster et al. [12], to encompass hierarchically-structured XML data. We present first the annotated XML data model, then illustrate the operation of queries on such data (for a fragment of XQuery) via several examples. Key properties of K-relations (in particular the natural analogue of the fundamental theorem 3.3) continue to hold in the generalized setting, which can be interpreted as evidence for the "robustness" of K-relations.

We fix a commutative semiring K and consider XML data modified so that instead of lists of trees (sequences of elements) there are *sets* of trees. Moreover, each tree belonging to such a set is decorated with an annotation $k \in K$. Since *bags* of elements can be obtained by interpreting the annotations as multiplicities (by picking K to be $(\mathbb{N}, +, \cdot, 0, 1)$), the only difference compared to standard XML is the absence of ordering between sib-

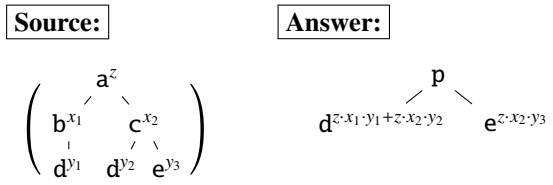


Figure 9: Simple `for` Example.

lings.[2] We call such data *K-annotated unordered XML*, or simply *K-UXML*. Given a domain $\mathcal{L}$ of *labels*, the usual mutually recursive definition of XML data naturally generalizes to *K-UXML*:[3]

- A *value* is either a label in $\mathcal{L}$, a tree, or a K-set of trees;
- A *tree* consists of a label together with a finite (possibly empty) K-set of trees as its "children";
- A finite *K-set of trees* is a function from trees to K such that all but finitely many trees map to 0.

In examples, we illustrate K-UXML data by adding annotations as a superscript notation on the label at the root of the (sub)tree. By convention *omitted annotations* correspond to the "neutral" element $1 \in K$.[4] Note that a tree gets an annotation only as a member of a K-set. To annotate a single tree, we place it in a singleton K-set. When the semiring of annotations is $(\mathbb{B}, \vee, \wedge, \mathsf{false}, \mathsf{true})$ we have essentially unannotated unordered XML; we write UXML instead of $\mathbb{B}$-UXML.

In Figure 9, two K-UXML data values are displayed as trees. The source value can be written in document style as: <a^z> <b^{x_1}> d^{y_1} </> <c^{x_2}> d^{y_2} e^{y_3} </> </>

where we have abbreviated leaves <*l*></> as *l*.

We propose a query language for K-UXML called *K-UXQuery*. Its syntax (see Foster et al. [12] for details) corresponds to a core fragment of XQuery [11] with one exception: the new construct `annot` k p allows queries to modify the annotations on sets. With `annot` k p any K-UXML value can be built with the K-UXQuery constructs. We use the following types for K-UXML and K-UXQuery:

$$t \quad ::= \quad \mathit{label} \mid \mathit{tree} \mid \{\mathit{tree}\}$$

where *label* denotes $\mathcal{L}$, *tree* denotes the set of all trees and {*tree*} denotes the set of all finite K-sets of trees.

In the rest of this section, we illustrate the syntax of K-UXQuery on K-UXML informally on some simple examples to introduce the basic ideas. We start with very simple queries demonstrating how the individual operators work, and build up to a larger example corresponding to a translation of a relational algebra query.

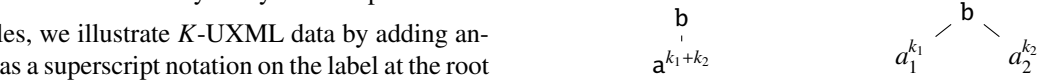
As a first example, define the following queries:

$$p_1 \stackrel{\text{def}}{=} \texttt{element}\ a_1\ \{()\} \quad \text{and} \quad p_2 \stackrel{\text{def}}{=} \texttt{element}\ a_2\ \{()\}$$

Each p_i constructs a tree labeled with a_i and having no children—i.e., a leaf node. The query `(`p_1`)` produces the singleton K-set in which p_1 is annotated with $1 \in K$. The query `annot` k_1 `(`p_1`)` produces the singleton K-set in which p_1 is annotated with $k_1 \cdot 1 = k_1$. We can also construct a union of K-sets:

$$q \stackrel{\text{def}}{=} \texttt{annot}\ k_1\ (p_1)\texttt{,annot}\ k_2\ (p_2)$$

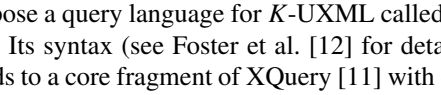
The result computed by q depends on whether a_1 and a_2 are the same label or different labels. If $a_1 = a_2 = \mathsf{a}$, then p_1 and p_2 are the same tree and thus the query `element b {`q`}` produces the left tree below. If $a_1 \neq a_2$, then the same query produces the tree on the right.



Next, let us examine a query that uses iteration:

```
p = element p { for $t in $S return
                  for $x in ($t)/* return
                    ($x)/* }
```

If `$S` is the (source) set on the left side of Figure 9, then the answer produced by p is the tree on the right in the same figure.[5] Operationally, the query works as follows. First, the outer `for`-clause iterates over the set given by `$S`. As `$S` is a singleton in our example, `$t` is bound to the tree whose root is labeled `a` and annotation in `$S` is z. Next, the inner `for`-clause iterates over the set of trees given by `($t)/*`:



It binds `$x` to each of these trees, evaluates the `return`-clause in this extended context, and multiplies the resulting set by the annotation on `$x`. For example, when `$x` is bound to the `b` child, the `return`-clause produces the singleton set `(`d^{y_1}`)`. Multiplying this set by the annotation x_1 yields `(`$d^{x_1 \cdot y_1}$`)`. After combining all the sets returned by iterations of this inner `for`-clause, we obtain the set `(`$d^{x_1 \cdot x_1 + x_2 \cdot y_2}$`,` $e^{x_2 \cdot y_3}$`)`. The final answer for p is obtained by multiplying this set by z. Note that the annotation on each child in the answer is the sum, over all paths that lead to that child in `$t`, of the product of the annotations from the root of `$t` to that child, thus recording *how* it arises from subtrees of `$S`.

Next we illustrate the semantics of XPath `descendant` navigation (shorthand `//`). Consider the query:

[2] For simplicity, we also omit attributes and model atomic values as the labels on trees having no children.

[3] In the XQuery data model, sets of labels are also values; it is straightforward to extend our formal treatment to include this.

[4] Items annotated with 0 are allowed by the definition but are useless because our semantics interprets 0 as "not present/available".

[5] Actually this query is equivalent to the shorter "grandchildren" XPath query `$S/*/*`; we use the version with a `for`-clauses to illustrate the semantics of iteration.

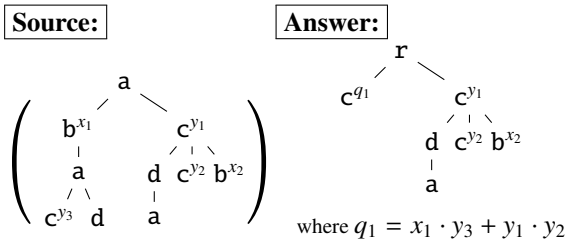


Figure 10: XPath Example.

$$r = \texttt{element r \{ \$T//c \}}$$

which picks out the set of subtrees of elements of `$T` whose label is `c`. A sample source and corresponding answer computed by r are shown in Figure 10. Foster et al. [12] defines the semantics of the `descendant` operator using structural recursion and iteration. It has the property that the annotation for each subtree in the answer is the sum of the products of annotations for each path from the root to an occurrence of that subtree in the source, like the answer shown here.

Now we turn to an example which demonstrates how K-UXQuery behaves on an encoding of a database of relations whose tuples are annotated with elements of K, i.e., the *K-relations* of Section 3. Consider again the relational algebra query from Figure 1c evaluated over the table of Figure 2, viewed as an $\mathbb{N}[X]$-relation.

Figure 11 shows the $\mathbb{N}[X]$-UXML tree that is obtained by encoding the relation hop, the corresponding translation of the view definition into $\mathbb{N}[X]$-UXQuery, and the $\mathbb{N}[X]$-UXML view that is computed using $\mathbb{N}[X]$-UXQuery. Observe that the result is the encoding of the $\mathbb{N}[X]$-relation of Figure 5. This equivalence holds in general: Foster et al. [12] give a systematic translation of positive relational algebra queries into K-UXQuery which agrees with the definitions of Section 3.

In a K-relation, annotations *only* appear on tuples. In our model for annotated UXML data, however, every internal node carries an annotation (recall that, according to our convention, every node in Figure 11 depicted with no annotation carries the "neutral" element $1 \in K$). Therefore, we have more flexibility in how we annotate source values—besides tuples, we can place annotations on the values in individual fields, on attributes on the relations themselves, and even on the whole database! It is interesting to see how, even for a query that is essentially relational, these extra annotations participate in the calculations. See Foster et al. [12] for an example.

6. FURTHER CONSIDERATIONS

In the few years since the publication of our paper with Tannen introducing semiring-annotated relations [19], there has been a flowering of work in the area. Various researchers have investigated different aspects of annotated data and provenance. We conclude this column with a brief overview of these works.

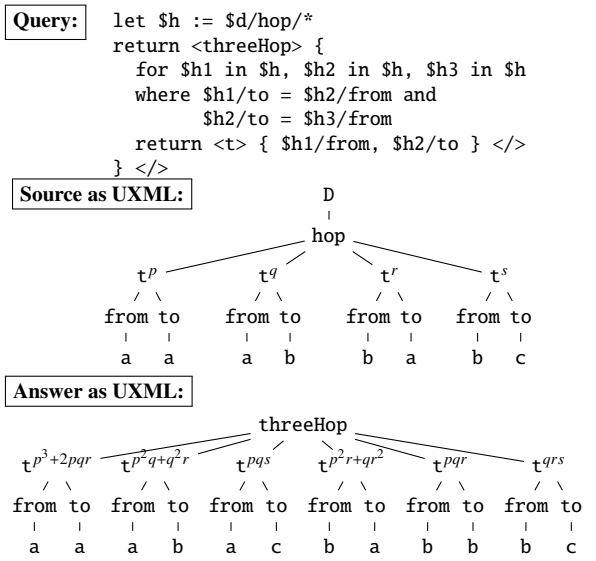


Figure 11: Relational (encoded) example

Recursive queries.

Apart from semantics for $\mathcal{RA}^+$ queries, in our paper with Tannen [19] we also gave semantics for recursive Datalog queries on K-relations. The high-level idea is that the provenance of a tuple in the result of a Datalog query is the sum over all its derivation trees of the product of the tags of the leaves of each tree. In general, however, a tuple can have infinitely many derivation trees. As a result, we focused on ω-continuous semirings, for which infinite sums can be defined. The most general such semiring, that can be used as the provenance model for Datalog queries, is the commutative ω-continuous semiring of **formal power series** with variables from X and coefficients from $\mathbb{N}^\infty$. Interestingly, even if the provenances of some tuples are infinite, they can be represented through a finite system of equations, whose solutions are the provenance expressions for each tuple. Moreover, the provenance graph we showed in the previous section can provide an alternative finite representation for the provenance of Datalog queries.

Query containment.

The introduction of annotations on relations presents new challenges in query reformulation and optimization, as queries that are semantically *equivalent* when posed over ordinary relations may become *inequivalent* when posed over K-relations. Indeed, this phenomenon was previously observed for the case of bag semantics [7, 22], where, e.g., adding a "redundant" self-join to a query actually changes the query's meaning. Green [15] studied containment and equivalence of (unions of) conjunctive queries under five difference kinds of provenance annotations—$\mathbb{N}[X]$, $\mathbb{B}[X]$, $\mathsf{Trio}(X)$, $\mathsf{Why}(X)$, and $\mathsf{Lin}(X)$—and established decision procedures and complexity characterizations in each case. Interestingly, the provenance hierarchy of Figure 6, which organizes provenance semir-

ings in terms of their information content, also turns out to organize them in terms of "stronger" to "weaker" notions of containment/equivalence. Kostylev et al. [26] nicely generalize the containment results by studying how these containment procedures *axiomatize* various classes of semirings.

Negation.

Semirings do not contain any operation that can capture the semantics of relational *difference*. Note that, e.g., for bag semantics, "regular" subtraction does not work for this purpose, because a tuple cannot have a negative multiplicity. For this reason, Geerts et al. [14] identified a class of semirings that can be extended with a *monus* operator, that captures the semantics of relational difference, and showed that the fundamental theorem holds for all such extended structures (called *m*-semirings). However, they also showed that the structure obtained by extending polynomials with monus is not the most general *m*-semiring (e.g., does not satisfy a factorization theorem similar to Theorem 4.4). Thus, they proposed using as a provenance model for $\mathcal{RA}^+$ with difference ($\mathcal{RA}^+(\setminus)$) the free *m*-semiring, for which there is a standard (though not very practical, from a systems perspective) algebraic construction.

Amsterdamer et al. [3] have recently raised questions about the suitability of *m*-semirings for capturing the provenance of $\mathcal{RA}^+(\setminus)$ queries, by identifying an identity of $\mathcal{RA}^+(\setminus)$ (namely $R \bowtie (S \setminus T) = (R \bowtie S) \setminus (R \bowtie T)$) that fails for some important *m*-semirings (e.g., PosBool(X) or the semiring of confidentiality policies). However, they have left the study of a provenance model satisfying all identities of $\mathcal{RA}^+(\setminus)$ as an open problem for future work, as they also deemed the standard algebraic construction for the free structure satisfying these identities of $\mathcal{RA}^+(\setminus)$ to not be very practical.

RDF/SPARQL.

Similar challenges, stemming from a form of negation, are posed when trying to specify the semantics of SPARQL queries over RDF data and capture their provenance. Indeed, recent work on SPARQL provenance [31] has showed that provenance polynomials, and other relational provenance models, can be used to capture the semantics of a positive fragment of SPARQL. However, in that paper the authors also argue that queries employing the SPARQL OPTIONAL operator, whose semantics involves a form of difference, cannot be captured by these models. Unfortunately, *m*-semirings are not suitable for this purpose, either, due to subtle differences between the semantics of relational and SPARQL difference. The specification of formal semantics and provenance models for SPARQL queries is the subject of ongoing work.

Z-relations.

The paper by Green et al. [17] introduced $\mathbb{Z}$*-relations*, relations annotated with (positive or negative) integer values, where the difference operator of $\mathcal{RA}$ has a natural interpretation using the inverse operation of the ring, and proposed $\mathbb{Z}$-relations as a useful device for representing data and updates (insertion or deletion of tuples) in a uniform manner. $\mathbb{Z}$-relations turn out to be surprisingly "well-behaved" with respect to query optimization, in that equivalence of $\mathcal{RA}$ queries is decidable under the semantics (in contrast to set or bag semantics, where the problem is undecidable). The paper also proposed a unified perspective on *view maintenance* (propagating the effects of source data updates to materialized views) and *view adaptation* (recomputing materialized views after their definitions are changed), based on optimizing queries using materialized views. Under $\mathbb{Z}$-semantics, a sound and complete algorithm for this problem is possible (again in contrast to set or bag semantics, where any sound algorithm for the problem is necessarily incomplete). Green and Ives [16] present a practical implementation of these ideas.

Aggregates.

Beyond $\mathcal{RA}^+$, Amsterdamer et al. [4] investigated aggregate queries, and found that capturing their provenance requires annotating with provenance information not just tuples, but also individual values within tuples. Then, they provided a semantics for aggregation (including group-by) on semiring-annotated relations, that coincides with the usual semantics on set/bag relations for min/max/sum/prod, and commutes with semiring homomorphisms. The resulting provenance expressions involve tensor products between semiring annotations and possible aggregation values.

Minimization and factorization.

Recent work has focused on alleviating practical challenges that arise in data management tools due to the size of provenance information, by proposing methods to reduce this size. In particular, Amsterdamer et al. [2] studied provenance *minimization*, and defined the *core* of provenance information, namely the part of provenance that appears independently of the query plan that is in use.[6] They also provided algorithms that, given a query, compute an equivalent one that realizes the core provenance for all tuples in its result, as well as algorithms to compute the core provenance of a particular tuple in a query result without re-evaluating the query.

Olteanu and Zavodny [29] considered *factorization* of provenance polynomials, which conceptually follows an approach akin to the relational provenance storage

[6] among plans that are equivalent under standard set semantics, not to be confused with containment and equivalence on annotated relations described above.

scheme described in Section 4.2, by representing polynomials as nested expressions. Indeed, such factorized polynomials of query results can be exponentially more succinct than their flat equivalents, by identifying and reusing parts of a polynomial that are common across multiple monomials. Moreover, they make explicit some of the structure in the query result. A downside of factorization is that the monomials of a factorized polynomial are not represented explicitly, but they can be enumerated (similarly to traversing a provenance graph backwards) with polynomial delay.

Recording schema mapping names in provenance.

In data sharing settings, such as addressed in the ORCHESTRA collaborative data sharing system [18], there is a need for provenance to also record the names of *schema mappings* (expressed as database queries) via which data are propagated. For this purpose, Karvounarakis [23] extended semirings with a set of unary functions, one for each mapping, and showed that they satisfy the fundamental theorem, as well as that the corresponding extension of provenance polynomials is the most general such structure. The provenance graph was similarly extended by labeling join nodes with the name of the corresponding mapping.

Querying provenance.

In Section 4.2 we explained how provenance graphs can be traversed to reconstruct provenance polynomials. In more complicated data sharing settings, involving nested and possibly recursive queries or schema mappings [18], it is also important to be able to isolate parts of the provenance of a tuple, e.g., relative to another derived tuple in the intermediate result of some other query, or only focus on parts of derivations involving certain mappings. The provenance query language ProQL [25] employs path expressions to simplify such operations, involving traversal and projection of parts of provenance graphs. Moreover, ProQL takes advantage of the factorization theorem for provenance polynomials, by supporting evaluation of provenance expressions (corresponding to matched graph parts) to compute annotations from various semirings.

Acknowledgments

We are grateful to Val Tannen and Nate Foster, our coauthors from the two papers [19, 12] on which much of this article is based. We also thank Zack Ives, whose ORCHESTRA project was the motivating context for the development of K-relations, and the Penn DB group and our colleagues at LogicBlox for many useful discussions.

7. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
[2] Y. Amsterdamer, D. Deutch, T. Milo, and V. Tannen. On provenance minimization. In *PODS*, 2011.
[3] Y. Amsterdamer, D. Deutch, and V. Tannen. On the limitations of provenance for queries with difference. In *TaPP*, 2011.
[4] Y. Amsterdamer, D. Deutch, and V. Tannen. Provenance for aggregate queries. In *PODS*, pages 153–164, 2011.
[5] O. Benjelloun, A. D. Sarma, A. Y. Halevy, and J. Widom. ULDBs: Databases with uncertainty and lineage. In *VLDB*, 2006.
[6] P. Buneman, S. Khanna, and W.-C. Tan. Why and where: A characterization of data provenance. In *ICDT*, 2001.
[7] S. Chaudhuri and M. Y. Vardi. Optimization of real conjunctive queries. In *PODS*, 1993.
[8] J. Cheney, L. Chiticariu, and W. C. Tan. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4):379–474, 2009.
[9] L. Chiticariu and W.-C. Tan. Debugging schema mappings with routes. In *VLDB*, 2006.
[10] Y. Cui, J. Widom, and J. L. Wiener. Tracing the lineage of view data in a warehousing environment. *TODS*, 25(2), 2000.
[11] D. Draper, P. Fankhauser, M. Fernandez, A. Malhotra, M. Rys, J. Simeon, and P. Wadler. XQuery 1.0 formal semantics. Available from `http://www.w3.org/TR/xquery-semantics/`, 12 November 2003. W3C working draft.
[12] J. N. Foster, T. J. Green, and V. Tannen. Annotated XML: Queries and provenance. In *PODS*, 2008.
[13] N. Fuhr and T. Rölleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *TOIS*, 14(1), 1997.
[14] F. Geerts and A. Poggi. On database query languages for K-relations. *J. Applied Logic*, 8(2), 2010.
[15] T. J. Green. Containment of conjunctive queries on annotated relations. *Theory Comput. Syst.*, 49(2), 2011.
[16] T. J. Green and Z. G. Ives. Recomputing materialized instances after changes to mappings and data. In *ICDE*, 2012.
[17] T. J. Green, Z. G. Ives, and V. Tannen. Reconciliable differences. *Theory of Computing Systems*, 49(2), 2011.
[18] T. J. Green, G. Karvounarakis, Z. G. Ives, and V. Tannen. Update exchange with mappings and provenance. In *VLDB*, 2007. Amended version available as Univ. of Pennsylvania report MS-CIS-07-26.
[19] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS*, 2007.
[20] T. J. Green and V. Tannen. Models for incomplete and probabilistic information. In *EDBT Workshops*, 2006.
[21] T. Imielinski and W. Lipski. Incomplete information in relational databases. *JACM*, 31(4), 1984.
[22] Y. Ioannidis and R. Ramakrishnan. Containment of conjunctive queries: beyond relations as sets. *TODS*, 20(3), 1995.
[23] G. Karvounarakis. *Provenance for Collaborative Data Sharing*. PhD thesis, University of Pennsylvania, July 2009.
[24] G. Karvounarakis and Z. G. Ives. Bidirectional mappings for data and update exchange. In *WebDB*, 2008.
[25] G. Karvounarakis, Z. G. Ives, and V. Tannen. Querying Data Provenance. In *SIGMOD*, 2010.
[26] E. V. Kostylev, J. L. Reutter, and A. Z. Salamon. Classification of annotation semirings over query containment. In *PODS*, 2012.
[27] W. Kuich. Semirings and formal power series. In *Handbook of formal languages*, volume 1. Springer, 1997.
[28] `http://www.logicblox.com`.
[29] D. Olteanu and J. Zavodny. Factorised representations of query results: Size bounds and readability. In *ICDT*, 2012.
[30] A. D. Sarma, O. Benjelloun, A. Halevy, and J. Widom. Working models for uncertain data. In *ICDE*, 2006.
[31] Y. Theoharis, I. Fundulaki, G. Karvounarakis, and V. Christophides. On provenance of queries on semantic web data. *IEEE Internet Computing*, 15(1):31–39, 2011.
[32] L. A. Zadeh. Fuzzy sets. *Inf. Control*, 8(3), 1965.
[33] E. Zimányi. Query evaluation in probabilistic relational databases. *TCS*, 171(1-2), 1997.

Handling Temporal Information in Web Search Engines

Edimar Manica
Campus Avançado Ibirubá, IFRS
Ibirubá, Brazil
II, UFRGS
Porto Alegre, Brazil
emanica@inf.ufrgs.br

Carina F. Dorneles
INE/CTC, UFSC
Florianópolis, Brazil
dorneles@inf.ufsc.br

Renata Galante
II, UFRGS
Porto Alegre, Brazil
galante@inf.ufrgs.br

ABSTRACT

The Web can be considered a vast repository of temporal information, as it daily receives a huge amount of new pages. Generally, users are interested in information related to a specific temporal interval. In the information retrieval area, researches have newly incorporated the temporal dimension to the search engines. This paper presents a comprehensive study that describes the evolution of search engines on the exploitation of temporal information. Research directions and future perspectives are also presented, considering the authors' point of view.

Keywords

Temporal information, temporal search engine, keyword search

1. INTRODUCTION

Web pages describe several topics, such as conferences, sports, politics and entertainment. The most of those events change over time. The SIGMOD conference, for instance, occurs every year. The World Cup occurs every four years. Dr. House series is a television medical drama displayed once a week. The database community has devoted extensive amount effort for indexing and querying temporal data in past decades [23, 13]. In recent years, the use of temporal expressions has emerged in Web search queries once Web documents also have temporal information, as we presented in previous work [14]. However, insufficient amount attention has been given to temporal queries on the Web, bringing a new challenge of the query processing on the Web: to take into account the temporal interval desired by the user.

Another important requirement for Web queries is that people are interested in latest information, e.g., "`Who are the last FIFA World Cup champions?`". Day after day, a huge number of new pages are posted on the Web. Most of these pages are available for a long time remaining a large repository of historical information. Moreover, users also can be interested in past (e.g., "`Which are the SIGMOD articles published in 2009?`" and "`What team won the World Cup in 2002?`") or future data (e.g., "`What is the weather forecast for next Monday?`" and "`What will happen in the Dr. House series in the next chapter of Tuesday?`").

The time concept can either help in recreating a particular historical period or describing the context of a document or collection. Furthermore, the time may be useful to improve the methods of ranking results by relevance [2]. The information retrieval area adopts this new insight by adding time dimension on ranking the results. The first initiative was to sort the results by considering the time, so that the latest results are shown on the top of the ranking through the concept of *freshness metric*. Here, we classify these search engines as First Generation. In the following, the concept of "filter results" is introduced in order to allow the user interaction by defining the notion of *temporal window of interest*. These search engines are here classified as Second Generation. Finally, search engines have begun to exploit the temporal information present in the Web documents content incorporating the *content-based temporal retrieval*. They differ from the previous generations since the first ones use only temporal information stored in Web documents metadata. We classify these search engines as Third Generation.

This paper presents a comprehensive study that describes the evolution of search engines on the exploitation of temporal information. We first discuss some special concepts used as base of our proposal. Then, we propose a new way for clustering search engines according their evolution features, categorizing them into three distinct generations, as already discussed before. A number of interesting research directions and future perspectives are also presented, considering the authors' point of view.

The rest of this paper is organized as follows. Section 2 discusses basic concepts used as groundwork in our categorization. Section 3 presents the study we have done in order to describe the evolution of search engines on the exploitation of temporal information and to propose the three generations. Section 4 points a number of interesting research directions that are open in relation to use temporal information in search engines.

2. BASIC CONCEPTS

A Web page can have different temporal information resources, which have been classified in three types [16], as described below.

- Content - a Web document's content can contain words and/or expressions with temporal meaning (e.g. "`today`", "`2011/10/16`", "`Christmas`")[1].
- URL Address - all public Web documents have at least one unique address. The different segments of an URL, namely host, path and search part, might be used as a source of temporal information. For instance, a current New York Times URL is structured in the following way - http: //www.nytimes.com /2011 /01 /26 /us/ politics/26speech.html?_r=1&hp. It is possible to derive the document's year, month and day of publication by parsing this URL.
- HTTP Protocol - the Hypertext Transfer Protocol (HTTP) is an application level protocol used to request and transmit hypertext documents and components between user applications and online servers. According to HTTP protocol, servers reply to each request, sending standard headers and, if available, the requested resource (body of the message). HTTP headers have a field that represents the date and time at which the resource was last modified (`Last-Modified` field). However, this field is not always available and may not return a valid date, mainly, due to incorrectly configured Web servers, or because the Last-Modified field indicates the date and time at which the origin server "believes" the variant was last modified[2].

Other temporal information that can be associated to a Web page by a search engine is the crawled date. Crawled date represents the date in which a Web page was indexed by the search engine. The temporal information presented in Web documents, such as a point in the time, an event or a period in the time, can be described in a conceptual level by a temporal entity. A sequence of tokens that represents an instance of a temporal entity is denominated temporal expression. Those temporal concepts have been introduced in [2] together with three temporal expression categories:

- Explicit - temporal expressions that directly describe an input in a timeline[3], such as an exact date or a specific year. For example, the expressions "`December, 2004`" or "`January 12, 2006`" in a text fragment are explicit temporal expressions and can be mapped directly into points in a timeline.
- Implicit - temporal expressions that need predefined knowledge (ontology describing temporal information, for instance) in order to be mapped into an input in a timeline. Holiday names and specific events are typical examples of implicit temporal expressions. For instance, the expression "`2005 Christmas`" needs to be mapped to "`December 25, 2005`".
- Relative - temporal expressions that represent temporal entities that can be only mapped into an input in a timeline in reference to an explicit or implicit temporal expression, or in reference to the moment the text has been written. For instance, the expression "`yesterday`" can only be mapped into a point in the time if we know the moment the document has been created.

The process of mapping temporal expressions into a standard format (so that it is possible to represent a point of a timeline in a standard way) is denominated temporal expression normalization. The search engines, which provide support to temporal information, index the normalized temporal expressions in order to allow a fast and consistent access to the temporal data.

On considering temporal expression in Web searches, an important concept highlights: temporal operator. A temporal operator defines temporal relations between two instants, two intervals or between an instant and an interval (*before*, *after* and *during*, for example). The temporal operators can modify the meaning of a temporal expression. For

[1]In this paper, the examples that represent a complete date are described in `YYYY/MM/DD` format.

[2]http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

[3]A timeline, also known as a chronology, is a linear representation of events in the order in which they occurred [2].

example, the following query "`2004 elections`" refers to the elections held in 2004. On the other hand, the query "`elections after 2004`" refers to the elections held after 2004, i.e., the temporal window of interest begins in 2005 and ends in the current year. This feature requires the appropriate temporal treatment of temporal operators. However, the current search engines manipulate temporal operators as a common keyword, i.e., they search for the occurrence of the keyword that represents the temporal operator in the indexed terms.

Allen [1] has defined thirteen temporal operators that can be used between intervals: `after`, `before`, `contains`, `during`, `equal`, `finished by`, `finishes`, `meets`, `met by`, `overlapped by`, `overlaps`, `started by` and `starts`. Figure 1 illustrates them. These operators are mutually exclusive, which characterize an issue that is not suitable for the context of Web searches because most queries are generic, which implies to use UNION in order to employ Allen's operators. For example, considering a query where the user wants pages that describe accidents that occurred between 2006 and the current year, and considering the Allen's operators, it would be necessary to use the operator `met by` together with the operator `after`, e.g. "accidents `met by` 2006 UNION accidents `after` 2006", to get the correct results, which probably would add a high degree of difficulty for most search engines users. Another example is a query in which a user wants pages about armed revolts that ended in 2005. Using Allen's operators, it is necessary to use the operator `finishes` together with the operator `finished by`, e.g., "`armed revolts finishes 2005 UNION armed revolts finished by 2005`". However, it would be more intuitive if there is a more general temporal operator specially projected to solve these cases. Probably, this generic temporal operator would be more useful than the operators `finishes` and `finished by` used separately, for example.

Figure 1: Allen's temporal operators

Manica et al. [14] have reported an analysis where it was verified that 33.96% of the queries (considering repeated queries) with temporal expressions have keywords that explicitly represent temporal operators. Considering just distinct queries, this percentage is 30.20%.

3. PAST AND PRESENT

In this section, we briefly describe an overview about temporal information management in some Web Search Engines, classifying them in three generations. Each new generation adds a new feature while keeping the old ones. The first generation of search engines introduces the concept of freshness metric. The second generation allows user interaction by defining the notion of temporal window of interest. Finally, third generation handles temporal information present on page content, supporting the content-based temporal retrieval. Third generation differs from the others since the previous generations use only temporal information stored in Web documents metadata.

3.1 First Generation - Freshness

The first generation of search engines is characterized by adding time while ranking the results through the freshness metric. The most recent pages are positioned at the top of the ranking[4]. The temporal information used in this case is the crawled date or the last modified date. T-Rank [4] is an algorithm which extends PageRank [17] to improve page ranking by exploring the freshness and activity of both pages and links. Traditional Search Engines, such as Google[5] and Yahoo![6] have adopted freshness metrics.

The strategy used in the first generation is to assume that most recently posted pages are most relevant to the user, since they mean to have the latest information. This assertion is typically valid for news, where the user generally wants to find novelty, since the oldest pages may have already been read (supposedly). However, this strategy does not benefit those users interested on historical information. For instance, a team X played a game on 2012/01/11 and another game on 2012/01/15. A user wants information about the first game (since he/she watched the last game, but lost the first one). Using a first generation search engines, the results at the top of the ranking refer to the pages with information about the game played on 2012/01/15. It is true once these pages have been created after the pages that have information about the game played on 2012/01/11, so they are more recent. Of course, it is possible to create a query that returns the pages about the game played on 2012/01/11 at the top of the ranking, using other metrics implemented by the search engines, as for instance, adding to the query a keyword that represents the opposing team, or the name of the stadium where the game has taken place. However, most users do

[4]Other metrics, such as page popularity, might also be considered. However, it is not the focus of this paper.
[5]http://www.google.com
[6]http://www.yahoo.com

not know how to create such a query. To solve this problem, the second generation of search engines adds the notion of temporal window of interest.

3.2 Second Generation - Temporal Window of Interest

The second generation of search engines defines the notion of temporal window of interest, where the users can specify the temporal interval that represents their interest. This temporal interval is defined in a specific field. The temporal information used in this case is the *crawled date*. Google, Chronica [7] and InfoSeek[7] are examples of second generation search engines.

Extracting the crawled date is a trivial task, since the search engines just need to obtain the current timestamp when storing the page on database. However, usually there is a gap between the page crawled date and the date to which the page content is related to. This gap mainly occurs in three situations: (i) when temporal information, present at the page content, is equal to the posting date but different to the crawled date; (ii) when the page content is about historical information; and (iii) when the page content has information about future. The following examples describe these situations.

The first situation can be described by considering, for instance, a page x posted on 2011/03/07, whose content is about games played on this date and with a few important links that point to it. Therewith, this page can be crawled on 2011/03/10 (i.e., days after the posting date). So, the search engines associate this date as the temporal information, although it is not the real temporal information of the page content.

The second situation can be exemplified by given a page posted, for instance, on 2011/12/14 with information about the FIFA 2002 World Cup and with several important links pointing to it. These features create a favorable situation for the page to be crawled on the same day it was posted. In this case, the temporal information associated to the page is 2011/12/14, although the page contains information about an event that occurred in 2002.

Finally, we describe the last situation by considering a page posted, for example, on 2011/05/20, with information about weather forecast for 2011/05/25 and with several important links pointing to it. It is another case where the page is crawled on the same day it was posted. The temporal information associated with the page is 2011/05/20 even though the page contains information about the future, i.e., 2011/05/25.

[7]http://www.infoseek.co.jp/

In order to solve the above problems, the third generation search engines focus is the exploration of temporal expressions present in the Web documents content.

3.3 Third Generation - Content-based Temporal Retrieval

The third generation of search engines exploits the temporal expressions present in the Web documents content in order to improve the search result quality supporting the content-based temporal retrieval. The search engines of this generation have two big challenges: (i) to extract the temporal expressions from Web pages content, and (ii) to define how to manipulate the temporal information. Therefore, before we describe the search engine, we present some temporal expression extraction tools, since they are a key feature of a search engine that manipulates temporal information. Section 3.3.1 presents the main tools for extracting temporal expressions while Section 3.3.2 shows some works we classified as third generation search engines.

3.3.1 Tools for Extracting Temporal Expressions

The current tools for temporal expressions extraction in Web documents use named entities extraction techniques to identify the temporal expressions. Some works propose the use of an XML document to store the annotated expressions [2]. TimeML [21] is an emerging standard for events and temporal expressions annotation. However, there are many tools for annotating temporal expressions that define their own form of annotation. Bellow, we present the main tools for temporal expressions extraction, and a brief comparison among them[8].

- ANNIE [3] is an open source extraction tool that is part of the GATE framework [6]. Besides temporal information, ANNIE extracts location, people, organization, sports information and so on. The extraction is performed from named entities. Some predefined entities are available, and it is also possible to define new ones through a rule-based language that is embedded in the tool. The output is an XML file with the annotations of the entities, identified in a specific language of the tool. The XML handling can be done through an API designed to be used with the tool (attached at the framework). ANNIE annotates explicit, implicit and relative temporal expres-

[8]It is important to notice that these tools are language dependent and most of them are specific to English Language.

Table 1: Temporal Expressions Extraction Tools

	POS	Language	Availability	Category	Normalization	Isolation
ANNIE	Yes	Own	Yes	E, I, R	No	Yes
GUTime	Yes	TimeML	Yes	E, I, R	Yes	Yes
PorTexto	No	HAREM Directives	No	E, I, R	No	Yes
Chronos	Yes	TIMEX2	No	E, R	Yes	Yes

Label: E: explicit I: implicit R: relative

sions. However, it does not perform the temporal expressions normalization.

- GUTime [10] is an open source tool designed to annotate temporal information. The GUTime requires the TreeTagger, which is a POS (*Part-of-Speech*) tagger that labels the words of a text with its morphosyntactic features, such as verb and noun. GUTime uses these labels in its temporal expression inference rules. The result of the annotation process is an XML document with the temporal expressions tagged according to the TimeML language. This tool annotates and normalizes explicit, implicit and relative temporal expressions. To normalize relative expression, such as today, tomorrow, yesterday, next month and last year, GUTime checks the local context in order to identify a reference point in time in which this temporal information are related to. Typically, the temporal reference point is the date of document publication.

- PorTexto [5] is a tool that recognizes temporal entity in Portuguese Language documents. The tool processes the document sentence by sentence, differently of other tools in which the text is processed word by word. The temporal expressions identification is performed by using expression patterns based on co-occurrences. These patterns are defined by a set of reference temporal words (PTR in Portuguese). A reference temporal word is a word that occurs in temporal expressions with at least two words. For instance, `year` is a reference temporal word because it occurs in the temporal expressions "`in last year`", "`in the year of 2009`" and "`a year ago`". If a sentence has a number, but it does not have a reference temporal word, then it is not considered a temporal expression, e.g., "`the product code is 2009`". The list of PTR is manually created. The extracted patterns are defined by regular expressions and stored in a file. It is possible to change the existing patterns and even include new ones. The output is an XML document with temporal annotations that follow the directives proposed by HAREM [19][9]. PorTexto annotates explicit, implicit and relative temporal expressions. However, it does not perform the temporal expression normalization.

- Chronos [15] is a tool designed to perform the recognition and normalization of temporal expressions. The text processing involves the extraction of tokens, a linguistic processing and the recognition of multi-words that is based on a list of 5,000 entries retrieved from WordNet[10]. After that, a set of approximately 1,000 rules is used to recognize temporal expressions and to extract information about them that are useful for the normalization process. Then, composition rules are performed to solve ambiguities when multiple labels are possible. The output is an XML document with temporal annotations in the language TIMEX2[11] [8]. Chronos annotates explicit and relative temporal expressions.

Table 1 presents a comparison among temporal annotations tools discussed in this section. In that table, we consider the following comparison items:

- `POS` - indicates whether the tool has linguistic processing.
- `Language` - indicates which language is used to annotate the temporal expressions.
- `Availability` - denotes whether the tool is available for download.
- `Category` - identifies which temporal expressions categories (E: explicit, I: implicit and R: relative) are supported by the tool, considering those described in Section 2.
- `Normalization` - indicates whether the tool normalizes the temporal expressions.

[9]HAREM is a joint assessment in the area of named entity recognition in Portuguese.
[10]http://wordnet.princeton.edu/
[11]TimeML is an extension of TIMEX2.

- `Isolation` - denotes whether each expression is singly evaluated or whether there is a process that seeks for patterns in all expressions of the page in order to disambiguate formats and gather information to improve the annotations.

Notice that the most of the tools uses some type of linguistic processing, which increases the cost of processing. Each tool uses a different language for annotating temporal expressions, although all languages are based on XML. Only GUTime and ANNIE are available for use. Most tools handle with explicit, implicit and relative temporal expressions. Only GUTime and Chronos perform the temporal expression normalization. All tools individually evaluate each temporal expression.

The growing availability of collections with annotated expressions allows the application of supervised machine learning techniques for the task of recognizing temporal expressions. ATEL [11] and Alias-i's LingPipe[12] are examples of these systems. The first one uses Support Vector Machine (SVM) while the other one uses Hidden Markov Model (HMM).

Several types of documents are available on the Web such as Web pages, XML documents, PDF documents, etc. The techniques for extracting temporal information may not be suitable for all types of documents. For example, the tools described above are not suitable for data-centric XML documents[13]. These documents rarely contain sentences with morphosyntactic elements, because generally they have only nodes with nouns. Therefore, tools that use linguistic processing are not proper for data-centric XML documents since they perform unnecessary processing. Moreover, the fact that these tools evaluate each temporal expression in isolation generates the loss of valuable information for the temporal expressions normalization. The process of grouping the terms according to the path that contains them and analyzing together all expressions of the same path provides more information to the normalization process. For example, the temporal expression format shown in line 4 in Figure 2 is ambiguous. It means that it is impossible to find out whether the temporal expression is related to "March 12, 2011" or "December 3, 2011". However, when checking the other values of the same XML path (`people/person/birth`) we can infer that the format of the path is `day/month/year`.

[12]http://alias-i.com/lingpipe

[13]XML documents are classified as data-centric when their data are structured. The name of the nodes typically represents semantic annotation.

```
01.  <people>
02.   <person>
03.     <name>XYZ</name>
04.     <birth>12/03/2011</birth>
05.   </person>
06.   <person>
07.     <name>TZY</name>
08.     <birth>25/03/2011</birth>
09.   </person>
10. </people>
```

Figure 2: Example of an XML document containing temporal data

TPI [14] is a third generation temporal search engine specifically designed for data-centric XML documents. TPI defines a temporal expressions tool that clusters the expressions according to their path in order to obtain information for the normalization process. This tool also has some heuristics that identify temporal intervals and dates that are structured in different elements.

3.3.2 Search Engines

In this section, we present some third generation search engines, and a brief comparison of their main features.

- TISE [12] indexes one temporal expression per page by selecting the temporal expression that better describes the events in the Web page. In the query, the temporal predicate is specified as an interval, which must be defined in a different field of that used for other keywords. The temporal query predicate is applied to the temporal expression indexed to the page. The temporal information is represented as an interval.
- TERN [22] indexes all the temporal expressions of a Web page. In the query, the temporal predicate is posed in the same field of the other keywords. The temporal predicate is applied to any temporal expression indexed to the page. The temporal information is represented as an instant.
- Pasca [18] proposes a temporal search engine for users who wants to find out when a particular event has occurred. It means that the user does not pose a temporal predicate in the query, but receives a temporal value as a result. Pasca indexes a temporal expression for each *temporal nugget*, creating a pseudo-document. A temporal nugget is a fragment of a sentence that notifies open domain facts associated with some entity. For example, "`Michael Jackson was`

`born on August 29, 1958.`". The temporal information is represented as an instant.

Table 2 presents a comparison among the third generation of search engines. We consider the following comparison items:

- `Predicate` - indicates whether the temporal predicate is embedded in the query or it is reported in a distinct field.
- `Label` - denotes whether the temporal information is represented as an instant or as an interval.
- `Temporal index` - indicates which temporal information is indexed.
- `Query type` - denotes the query type: (i)`Temporal Selection`, temporal predicate is used to filter the query result, and (ii) `Temporal Output`, the user wants to know in which time a certain event has happened.

Notice that TERN is the only search engine that allows the temporal predicate to be posed in the same field as the other keywords in the query. This feature requires an extra step for identifying and normalizing the query temporal expressions. However, it simplifies and accelerates the creation of the query, since it is not necessary to fill several distinct fields. TISE is the only search engine that represents the temporal information as an interval. The representation as an interval is wider since it allows, for example, in the query: "`preside Senate between 2001 and 2003`", to express that "2002" is also valid.

Each search engine indexes different temporal information. Indexing only one temporal expression per page (TISE) causes the loss of relevant temporal information. Indexing each temporal nugget associated with a temporal expression (Pasca), instead of indexing all the temporal expressions in a page (TERN), has the advantage of associating a temporal expression only with the terms that are close in the page content. Most search engines allow temporal selection queries. TISE, Pasca and TERN use techniques of extraction tools for identifying and normalizing temporal expressions in order to index the temporal data in a consistent and standardized way.

The main challenge is to treat the temporal expressions formulated in a query. This issue is new and there is no appropriated treatment for temporal operators (as discussed in the end of Section 2). Notice that the temporal predicate in TISE is posed in a fixed field as an interval. PASCA has no temporal predicate. TERN has the embedded predicate in the query, but does not address temporal operators.

4. FUTURE DIRECTIONS

A number of interesting and important research directions are opened when handling temporal information in web search engines. We point out some of them below.

1. **Temporal Information Weight** - temporal information is an important feature to be considered in the ranking of Web pages. However, there are other very relevant metrics to be used, for example, the popularity. The challenge for future research is the weight of temporal information comparing to other metrics. Furthermore, it is necessary to consider the temporal proximity. For example, in a query where the user wants information from the last 3 days, information from four days ago may be relevant. As already discussed in several work such as [9, 20], we know that in the most of the Web queries, users do not know exactly what they are looking for or they do not know how to properly express their need.
2. **Embedded Temporal Query** - most temporal search engines has an additional field where the user specifies the desired time period. However, handling temporal operators is an open field. Consider, as an example, a query where the temporal information is posed in the same field of other keywords. How to find the temporal operator? The temporal operators define temporal relationships, such as "after" and "before". These operators can be explicitly specified, e.g., "`President after 2000`"). Temporal operators also may be implicit, as for example in the query: "`military regime 64 84`". In this query, the temporal relationship can be "equal", i.e. the user wants to find pages having information about military regimes that started in 1964 and ended in 1984. On the other hand, the temporal relationship can be "intersection", i.e. the user wants to find pages on any military regime that happened between 1964 and 1984. The identification of implicit temporal operators requires the semantics and context analysis of the query.
3. **Index** - the index structure of a temporal search engine should consider the temporal dimension. A traditional search engine uses a

Table 2: Comparison among Third Generation Search Engines.

	Predicate	Label	Temporal Index	Query Type
TISE	Isolated	Interval	One Timex per page	Temporal Selection
TERN	Embedded	Instant	All timex in the page	Temporal Selection
Pasca	NA	Instant	One timex per temporal nugget	Temporal Output

Label: NA: Not Applicable Timex: Temporal Expression

traditional inverted index to store the terms. Usually, a list of postings (documents identifiers and pre-computed scores) per term is used, which is a scalable technique for Web search engines. However, adding the temporal dimension, the inverted index would contain the postings for all kind of temporal information. This problem increases when we consider multi-word queries. Unless the flexibility of multi-word query in the pre-identified interest is restricted, it requires a positional index. In this index, each word list contains postings for each occurrence of the word in each document. A new challenge is to add temporal dimension in order to optimize the index structure [23].

4. **Temporal Expressions Normalization** - although there are several proposals for identification, extraction and normalization of temporal expressions, there are still several gaps, such as format disambiguation and temporal interval identification. For example, having "`12/03/2011`" as a temporal expression, we can not guarantee that this temporal format is not ambiguous, since its format can be `day/month/year` or `month/day/year`. How to find the correct format? Another example, considering the identification of temporal intervals, is the sentence "`In 2004, Johny started as director of Tempo company, directing it until 2007`". This sentence has a temporal interval that begins in 2004 and ends in 2007. Thus, the search engine must have the knowledge that in 2005 Johny was also the director of the company Tempo. How to identify temporal intervals in Web pages?

5. **Temporal Ranking Queries** - the database community has devoted extensive efforts in order to index and query temporal data [23, 13]. However, insufficient attention has been given to queries with temporal ranking. For example, given any time instance t, the user could want to know about the top-k instances at the time t related to some score attribute. Ranking queries within a temporal interval rather than just at one time instance also is an open field.

6. **Temporal Evidence** - the Web is a highly dynamic environment, with significant updates occurring weekly. In this scenario, temporal evidence might be obtained from the temporal evolution of the content and structure from each individual document, and from the whole Web. This dynamic behavior creates another challenge that is: how can we use this temporal evidence to improve information retrieval? Nunes [16] discusses some challenges in this context.

5. CONCLUSION

Time is an important dimension for any application. Realizing the value of temporal information for information retrieval, researchers have begun to incorporate this dimension in Web search engines to improve their ranking mechanisms. The first initiative, and still used today, is to put the most recent pages in the top of the result. After that, some proposals have arisen with the idea of filtering the results, considering temporal intervals according to the page crawled date. Finally, search engines have been proposed in order to exploit the temporal information present in the contents of Web pages and/or the queries. This article has presented a set of search engine proposals and their mechanisms to incorporate temporal information treatment.

Finally, we suggested some challenges for future research, such as: (i) the importance of defining a temporal information weight to incorporate this feature in the ranking algorithm; (ii) the requirement for performing appropriate treatment of temporal operators in embedded temporal queries and; (iii) the necessity of modifying the traditional inverted index to aggregate a temporal dimension considering different temporal information resources (last-modified date, crawled date and temporal expressions presented in the Web page content).

6. ACKNOWLEDGMENTS

Work partially funded by the CNPq Research Grant (Process nr. 307992/2010-1. PQ 2010) and INCT (Process nr. 573871/2008-6).

7. REFERENCES

[1] J. F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983.

[2] O. Alonso, M. Gertz, and R. A. Baeza-Yates. On the value of temporal information in information retrieval. *SIGIR Forum*, 41(2):35–41, 2007.

[3] ANNIE. Open source information extraction, 2010. <http://www.aktors.org/technologies/annie/>.

[4] K. Berberich, M. Vazirgiannis, and G. Weikum. T-rank: Time-aware authority ranking. In S. Leonardi, editor, *WAW*, volume 3243 of *Lecture Notes in Computer Science*, pages 131–142. Springer, 2004.

[5] O. Craveiro, J. Macedo, and H. Madeira. Use of co-occurrences for temporal expressions annotation. In *SPIRE*, volume 5721 of *LNCS*, pages 156–164. Springer, 2009.

[6] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. A framework and graphical development environment for robust nlp tools and applications. In *ACL*, pages 168–175, 2002.

[7] D. Efendioglu, C. Faschetti, and T. J. Parr. Chronica: a temporal web search engine. In *ICWE*, pages 119–120. ACM, 2006.

[8] L. Ferro, I. Mani, B. Sundheim, and G. Wilson. Tides temporal annotation guidelines - version 1.0.2, 2001. MITRE Technical Report MTR 01W0000041. McLean, Virginia: The MITRE Corporation. June 2001.

[9] M. Franklin, A. Halevy, and D. Maier. From databases to dataspaces: a new abstraction for information management. *SIGMOD Rec.*, 34:27–33, December 2005.

[10] GUTime. Adding timex3 tags, 2010. <http://www.timeml.org/site/tarsqi/modules/gutime/index.html>.

[11] K. Hacioglu, Y. Chen, and B. Douglas. Automatic time expression labeling for english and chinese text. In A. F. Gelbukh, editor, *CICLing*, volume 3406 of *Lecture Notes in Computer Science*, pages 548–559. Springer, 2005.

[12] P. Jin, J. Lian, X. Zhao, and S. Wan. Tise: A temporal search engine for web contents. In *IITA '08*, pages 220–224, Washington, USA, 2008. IEEE Computer Society.

[13] F. Li, K. Yi, and W. Le. Top- queries on temporal data. *VLDB J.*, 19(5):715–733, 2010.

[14] E. Manica, C. F. Dorneles, and R. Galante. Supporting temporal queries on xml keyword search engines. *JIDM*, 1(3):471–486, 2010.

[15] M. Negri and L. Marseglia. Recognition and normalization of time expressions: Itc-irst at tern 2004, 2004. Technical report, ITC-irst, Trento.

[16] S. Nunes. Exploring Temporal Evidence in Web Information Retrieval. In A. MacFarlane, L. Azzopardi, and I. Ounis, editors, *BCS IRSG Symposium Future Directions in Information Access (FDIA 2007)*, pages 44–50. BCS IRSG, BCS IRSG, August 2007.

[17] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[18] M. Pasca. Towards temporal web search. In *SAC*, pages 1117–1121. ACM, 2008.

[19] D. Santos, C. Freitas, H. G. Oliveira, and P. Carvalho. Second harem: New challenges and old wisdom. In *PROPOR*, volume 5190 of *LNCS*, pages 212–215. Springer, 2008.

[20] J. Teevan, C. Alvarado, M. S. Ackerman, and D. R. Karger. The perfect search engine is not enough: a study of orienteering behavior in directed search. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '04, pages 415–422, New York, NY, USA, 2004. ACM.

[21] TimeML. Markup language for temporal and event expessions, 2010. <http://www.timeml.org>.

[22] M. T. Vicente-Díez and P. Martínez. Temporal semantics extraction for improving web search. In *DEXA Workshops*, pages 69–73. IEEE Computer Society, 2009.

[23] G. Weikum et al. Longitudinal analytics on web archive data: Its about time! In *5th Biennial Conference on Innovative Data Systems Research (CIDR2011)*, 2011.

Ryan Johnson: recipient of the 2012 ACM SIGMOD Jim Gray Dissertation Award

by Marianne Winslett and Vanessa Braganholo

Ryan Johnson

http://www.cs.toronto.edu/~ryanjohn/

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 Phoenix, site of the 2012 SIGMOD and PODS conference. I have here with me Ryan Johnson, who is the recipient of the 2012 SIGMOD Jim Gray Dissertation Award for his thesis entitled "Scalable Storage Managers for the Multi-Core Era". Ryan's advisor was Anastasia Ailamaki. His PhD is from Carnegie Mellon, and he is now a professor at the University of Toronto. So, Ryan, welcome!

What was your dissertation about?

Well, the short version is making databases run really fast on modern hardware. Especially changing modern hardware, because we are in a time where designs are under huge flux and the things that were problems let's say every ten years, are now problems basically at every hardware generation. So there is a lot of work for the database engine to keep pace and to insulate the rest of the database community from all of these disruptive chambers of the lower levels.

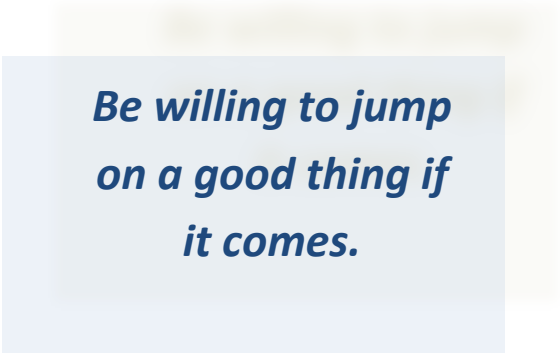
So what kinds of things do you have to change in the database core to keep up with the greater

parallel processing that we are now seeing?

Honestly, you have to be willing to change about anything, though you try to change as little as possible. The usual culprits are things that were centrally managed in the past. Things like locking and logging are huge culprits because they are central points of design. And so finding ways to either distribute those, or make them at least act like they are distributed without giving up the semantics, is the key to getting the scalability we need.

What does it mean for locking to be distributed?

So, by design, the database engine wants a central point where we can find the state of the world. And locking is designed to give you that, for the semantics of the applications that rely on it. So the application can say "I locked this value, nobody else is going to change this value". What, exactly, that means is mostly application dependent. The database's job is just to enforce the lock itself. The semantic of this globally visible, you know, one state of the world for each lock, means that you kind of need the central point, but it turns out that usually people aren't changing the protected value. So the trick was to identify when the bottleneck comes out of read-only values, and then play dirty tricks under the hood. Semantically, at the visible level, the system is still exactly the same as before, but under the hood, the lock protocol has been spread out so that these hot spots no longer arise. In practice, that works really well because a database application where people are fighting over updates doesn't scale for other reasons anyway.



Do you use a technique where you sometimes abort transactions later on because they had a conflict that you detected later on?

So that is standard practice for database engines, I didn't really change that part. What I was looking at was when a database says "I want to protect this from updates by other transactions: I'm only going to read, I just don't want the rug pulled out from under me while I am doing it". And everybody does this, it turns out. They say, "lock the database so that nobody deletes the database", "lock this table so that no one deletes this table", and it's those locks that were the bottleneck. Because everybody is doing it! And the time it takes to find out "is the lock compatible?" was long enough to create a bottleneck. So the trick was to say, since everybody does it, we'll assume that whoever is next will do it, we'll just keep the lock. It's read-only, it's not like anything has changed. The agent thread that was executing a transaction will just hold onto the lock, and hope that the next transaction can use it. And 90+% of the time, we're right, and now you've completely cut the [central] lock manager out of the picture, and so it is no longer involved.

Was there something for logging that you had to do?

Logging was a different one. Locking was theoretically parallel but in practice it wasn't because of the low level implementation. Logging is the opposite. It's supposed to be serial, right? We

want the one global history that Mohan gave us [with ARIES], so we can say the state of the world from all viewers is the same. But under the hood, we really want that to be parallel so that more than one thing can happen at a time. The trick there was to exploit a technique called linearizability, which essentially says we can play tricks with time. And so everyone will agree in the end on what the order was, but it's not necessarily the order that things actually happened in. But since we all agree, that's okay. Doing that relaxed the constraints on the implementation enough that we could do some things under the hood to get the parallelism that we needed. So at this point, you're going to run into other problems before the log becomes a problem again.

Has your dissertation work had impact?

It has. So first of all, the storage engine [Shore-MT] is in use by several researchers at various institutions, academic and industrial, and so that has been really nice. We debated whether to open source it or not, and in retrospect, it was a really good idea. People are using it and it is helping their research. Also, developers for the various open source databases have mentioned using various things [from my dissertation work]. I talked with people from MySQL, people from Berkeley DB, recently the PostgreSQL people talked about picking up the logging stuff, and so it has been really cool to see people actually interested in the work and trying to move forward with it.

You have to know why things are happening, and not only why they're happening, but why they didn't work, and why they did.

What about companies?

The commercial database vendors don't talk an awful lot about what's under the hood. I am in active talks with both Oracle and IBM, so they are interested in the kinds of things I do, but whether they are using precisely what I've already done, I don't know.

It's amazing to me to see such a core traditional database thesis being honored in this way because I discovered that the young people in our field, most of them, they don't know how a database works. Indexing, maybe. Read and write locks, yes. But anything beyond that, they have no idea. So do you think this is a problem, or is it okay because the field has expanded so much, people can't all know where we started?

This is a tricky one. It becomes a challenge sometimes because you submit all of these systems in papers, and don't necessarily get an audience who has any idea or interest in what it was [you did]. In the end, I decided this means that we're being successful. A small handful of people can make these problems go away, and nobody else has to worry about them. But it doesn't always work, right? We have to have some guidance from the people up higher of what they need so we can provide it. They need guidance from the people down lower about what's just not going to work, to shift their expectations a little bit. So the database engine is really this two sided thing:

there's the level that faces the user above, and there's the level that faces the hardware below. I'm on the hardware facing side, but we still have to talk to each other.

Is there any advice that you'd like to share with our readers or viewers who are working on their PhDs right now?

One final thing that I really became more aware of as got toward the end of my PhD was that you have to be willing to cross boundaries into nearby disciplines. It is not enough to be an expert in one narrow little tiny slice because it is all mined out, to be honest. Database engines are not new things, we've had them for 50, 60 years, and so if you want to really be able to solve these problems, you've got to be willing to bridge gaps between nearby things, and very often it turns out the problem actually isn't in the database engine. Some of the work that I enjoyed the most was actually going out into the operating system, or dealing with other nearby things, or talking to the SPAA community about distributed algorithms. Those things are what really enrich the research when you are able to go across boundaries and not be scared of new material.

Is there something you now know that you wish you had known earlier in your research career or perhaps during your job hunt that you can share with our younger readers and viewers?

So, I'll rephrase that a little, because I had a great advisor, who taught me early on things that I am really glad that she taught me. One of them was that you always want to understand what's going on. You get this result, and there is some hiccup or some bump in the data that you don't understand, you don't ever let that slide. You have to know why things are happening, and not only why they're happening, but why they didn't work, and why they did. Very often we get people with results were they say "we turned this knob and performance went up, let's declare victory and go home". But we don't know why it worked. We don't know how long it is going to work, or whether we just got lucky. So if you can point to things besides just performance and say, "look, I have analyzed the system and here are reasons x, y and z why we know this problem is gone and it's not going to be back". That gives you a much tighter control over your results because now you know why they are doing what they are doing. And that leads to whole new areas. One of my papers came from "why in the world would that happen that way?". It would have been a huge mistake to let that [question] slide.

Other things for the job hunt, I had a kind of odd job hunt because I only applied to one school. It was a surprise thing, and I said, "I'll try it, and if it doesn't work, I'll go on the job market next year". And it turned out I loved the place. So one thing I would say is "be willing to jump on a good thing if it comes". Don't keep searching after you've found something that is good. Sometimes we get stuck on this "I have to have the absolute best" and we let so many good things go by that we look back and regret it. So I am really glad that I have been able to jump on things early and then look back and say, "yeah that was actually a good idea".

Excellent advice. Thanks so much for talking with me today.

You're very welcome.

dbTrento: The Data and Information Management group at the University of Trento

Themis Palpanas
University of Trento, Italy
themis@disi.unitn.eu

Yannis Velegrakis
University of Trento, Italy
velgias@disi.unitn.eu

1. INTRODUCTION

The dbTrento group was established in 2006 by Profs Themis Palpanas and Yannis Velegrakis. Since then it has steadily grown into a fully functioning group with (currently) 17 members. It is located in Trento, a beautifully preserved historic town in the Dolomite mountains, which hosts one of the 6 ICT Labs of the European Institute of Innovation and Technology (EIT), and aims to become a reference research and technological center in Europe. The VLDB 2013 conference is being organized by dbTrento, its founders serving as the General Chairs. The mission of the group is to conduct high level research on different aspects of large scale data and information management. The following sections provide a high level description the work in these areas, which has also led to 3 Best Paper awards.

2. ADVANCED QUERY ANSWERING

[Semantic-based Keyword Search] Keyword search is becoming the de-facto mechanism for querying data [19], since it does not require knowledge of the full semantics or their organization in the repository, neither knowledge of some complex query language. For this reason, there has been enough work on querying structured (mostly relational) data through keywords. These works are typically based on an index that is built in advance, and which supports at run time the mapping of keywords to database structures. This index requirement makes these approaches difficult to apply when there is no prior access to the data, a common situation that occurs in integration system or on the web where the sources are autonomous and allow access to their data only through web interfaces or wrappers.

Collaboration with researchers from the University of Modena and the University of Zaragoza, has resulted into Keymantic [7], an engine for answering keyword queries over relational data that uses only the metadata provided by the database and some auxiliary information that is freely available on the web. Keymantic is using this information in order to understand the semantics of the keywords [6] and discover the best matching of these keywords to database tables, attributes or values. It does so by using an adapted version of the Hungarian algorithm. The discovered matches are combined to form SQL queries and returned to the user ranked in decreasing older of the likelihood that they represent the intended user query semantics. The KEYRY [9] is a version of Keymantics that uses a Hidden Markov Model, instead of the Hungarian algorithm, to make the predictions [8]. Apart from query answering, Keymantic can be used in data exploration. In particular, given a keyword query it can return structured queries exposing the structures of the data repository that may be related to the keyword query semantics. This can be applied not only on relational data but also on data of graph structure [10].

A lot of work within the group has been devoted to the management of entities that form the basis of Dataspaces and of Semantic Web Data [19]. One of these works is the discovery of the entities that best match a specification expressed in a keyword query. By exploiting attribute frequencies found in query logs, a classifier is trained to predict the intended semantics of the keyword query and to construct the answer as a ranked set of entities, with the most prominent at higher positions [35].

[Approximate Query Answering] Documents, or semi-structured data in general, provide a great deal of information, but their lack of schema makes information discovery a challenging tasks. Together with the University of Bozen-Bolzano and the University of Alberta, we have created TASM [3], a system that allows approximate query answering on large XML documents. It is based on the prefix ring buffer that allows the pruning of all the subtrees in the document above a threshold in a single postorder scan of the document, leading into an algorithm that depends only on the size of the query [4]. This work won the Best Paper award in ICDE 2010.

[Managing Evolution] The dynamic nature of the data has been realized from the very first days of data management. However, work has mainly focused on value

updates, offering the ability to query the data at different points of time. These approaches did not support the representation of relationships between different structures that model the same real world object at different points in time, which in turn made hard the realization of the evolution phases through which the object has gone through. Furthermore, even if such a modeling is made, it is not guaranteed that it will match the evolution model on which a query is constructed. For instance, a database may contain different instances of Germany from different phases of its history, i.e., as an empire, as the pre-war country, as East and West, and as a unified country. If these are different entities, a query asking for leaders of Germany, where Germany refers to the general concept of the German nation, will not be possible to answer.

With this in mind, our members have developed a mechanism that allows the evolution information to be included explicitly or implicitly [47, 45, 46, 42] in the data, forming an evolution graph. Apart from modeling and querying evolution [44], the system allows the answering of queries formed with a different evolution granularity in mind than the one of the repository [15]. This is achieved by performing at runtime merges of structures representing different evolution phases of an object. Achieving the minimum required such merges, boils down to efficient discovery of Steiner Forests which we solve using dynamic programming [14].

3. INFORMATION INTEGRATION

A long chapter in the dbTrento research agenda is related to Data and Information Integration from multiple disparate heterogeneous sources. Much of this work is based on entities as the basic data unit, driven by their recent popularity and expressiveness. Many of our developments were fundamental components in the OKKAM project that aims at the creation of an infrastructure for global identifiers for every web object [12, 5, 33, 27, 39] Our collaborations with colleagues from the Semantic Web community and our participation in many projects from that area has revealed numerous challenging data management issues, that were recently summarized in a related tutorial [24].

[Blocking Techniques] A basic task in every integration effort is Entity Resolution (ER), i.e., the ability to identify whether two pieces of information represent the same real world entity and then merge them into a single representation. This is inherently a quadratic task, requiring pair-wise comparisons among all objects in the collection. In order to scale ER to the volume of Web Data, blocking techniques are typically employed. However, most of the blocking techniques rely on schema information and are inapplicable to the highly heterogeneous settings of Web Data.

In collaboration with the L3S Research Center, we have proposed a novel framework consisting of two orthogonal layers (the effectiveness and efficiency layers), and showed how all blocking methods for highly heterogeneous data spaces map to this framework [41]. In addition, we have proposed several new techniques for improving the performance of ER [40, 41], namely, redundancy elimination, attribute-agnostic blocking, attribute clustering blocking, comparison pruning, and comparison scheduling, which all together offer significant time-performance improvements at a negligible cost on effectiveness. An interesting research direction would be to investigate the use of cloud computing and resource management technologies [28], in order to further improve the performance of the proposed techniques.

[On-the-Fly Entity Resolution] Traditional entity resolution approaches compute some similarity score between entities to decide whether they represent the same real world object. However, it is not clear what value of that score is high enough for such a decision, especially across different score computation methods. Furthermore, two representations may be seen as one in some cases and as independent in another, making the decision on the merging an application or query dependent.

With this is in mind, we have decided to take a radical approach and in collaboration with researchers from the L3S Research Center and the Technical University of Crete, we have developed LinkDB [26], a probabilistic linkage database system. We have thought that since it is hard to decide on whether a score is high enough to make a decision, we can postpone the decision until when needed. As such, we run entity linkage algorithms on our data but instead of doing any merging based on the results, we store the computed similarities in the repository alongside the data. At run time, when the user query is known, the system investigates what merges can be made and generate the answers the user query by evaluating it in a virtual database resulting from the merging of these entities [25]. An important feature of the system is that only the merges that affect the query results are taking place and not all the possible merges in the database. Another important feature is that the user may see results that are not in the database, but inferred from the stored data through on-the-fly merges.

[Schema and Data Mapping] A traditional topic of interest in the group is data mapping, i.e., the ability to associate data in one format with data in another. The work is rooted in our past participation in the development of Clio [20], a schema mapping tool from IBM Almaden. Clio was based on relational and semi-structured data, but we have ported this experience into

the development of mappings between entity repositories and ontologies. These new applications were developed in Papyrus [29], an EU project aiming at the creation of availability of content of one discipline to a different discipline, allowing people from the first, e.g., historians, to query data from the other, e.g., news, even if the latter uses a different terminology, structure [13], or language [52].

[Benchmarking] Evaluation is a fundamental step of every scientific finding, since it allows comparison with similar products and leads to informative decisions. Unfortunately, for many relatively novel areas like schema mapping and entity management, there is no globally accepted evaluation methodology. Researchers or vendors use their own tests and metrics leading into a blurred environment among similar products, developments or findings. Our group has spent a significant amount of effort into studying the problem and developing complete, consistent and principled evaluation methodologies that were recently presented in a tutorial [11]. Among these works is STBenchmark [1], a benchmark co-developed with the University of California Santa-Cruz, for evaluating mapping systems. It consists of a collection of test cases that can be used to measure the capabilities and limitations of a mapping system in terms of expressiveness and flexibility. Furthermore, it can dynamically generate of test cases at different levels of complexity and size, allowing the evaluation of the mapping systems in terms of scale [2]. In the same spirit with STBenchmark we have developed EMBench[1]. It is based on the same principles but it is designed for evaluating entity matching systems and can be a useful complement to the test cases provided by the OAEI initiative.

[Updates] Integration Systems have traditionally been considered read-only, mainly due to the fact that the system had no control over the data stored in the individual sources. Nevertheless, many times the integration reveals information not original available by the individual sources, thus, it may require that updates be issued on the integrated data. These updates have to be propagated to the sources. In collaboration with the AT&T Research Labs have studied ways to implement this goal [30, 46] and overcoming the difficulties that the view update theory is posing on the aspect.

4. STRUCTURED DATA ANALYTICS

[Data Stream Processing] The availability and use of (various types of) sensor networks have generated a lot of research interest. A major part of this effort has concentrated on how to efficiently collect and analyze the streams of sensed data. The dbTrento group has been working on several problems related to streaming data, ranging from data collection and representation, to data management, and analysis. Much of this work is also relevant to wireless sensor networks, and has been described in a recent survey of the area [37].

We have recently proposed a data-driven acquisition technique based on a linear model, DBP [43], that reduces the communication costs while mitigating the problems of noise and outliers. Relative to previous approaches, it can be much more efficiently implemented on resource-scarce nodes, and provides accuracy guarantees on the reported sensor measurements. Our work has shown that in the case of wireless sensor network deployments, further advances of the data management techniques would have little practical impact on the system lifetime [43]. Instead, improvements are more likely to come from radical changes at the routing and MAC layers, where new, data-aware protocols need to be designed. This work won the Mark Weiser Best Paper award in PerCom 2012.

The sheer number and size of the data we need to manipulate in many of the real-world applications dictates in several cases the need for a more compact representation than the raw data. We have developed novel, *amnesic* data approximation techniques that represent the most recent data with low error, and are more forgiving of error in older data, for arbitrary user-specified amnesic functions [38]. These techniques are incrementally maintainable, and are applicable to both landmark and sliding windows.

Several of the applications that consume streaming data, possibly from multiple sources, have high processing requirements over a significant portion of these data. We have developed a framework targeted to such applications, which aims to approximate in an online fashion multi-dimensional data series distributions [50]. This framework is adaptive, requires no a priori knowledge about the distributions of the sensed values, and it operates in a distributed fashion. We have demonstrated the applicability of the above framework in addressing two diverse and demanding problems, namely, identification and tracking of homogeneous regions [49], and outlier detection [50].

In a similar setting of multiple data stream processing in a network of nodes, we have proposed a technique for processing continuous queries that optimizes for the *profiled input throughput* that is focused on matching the expected behavior of the input streams [48]. We have also separately considered the problem of streaming sub-space clustering for high-dimensional spaces [36].

The efficient detection of frequent items in data streams is another interesting problem with many applications across domains. In this context, we have

[1]http://db.disi.unitn.eu/pages/EMBench/

performed a comprehensive comparative analysis of the available solutions, leading to several insights [31], and we have proposed a solution to the problem of finding recent frequent items in *ad hoc* windows in the past [51].

[Learning in Data Streams] Data streams can also carry information (e.g., user preferences) useful for learning algorithms. In this context, we have proposed a novel approach that can combine the content (descriptive aspect) and the type (directly quantifiable, or binary aspects) of the information instances, and studied the learning curves of the algorithms under different random information shifts [21]. This work won the Best Paper award in ADAPTIVE 2009.

Building on this work, we subsequently proposed an analytic model that describes the effect of the memory window size on the average prediction performance of a learning system, regardless of its underlying algorithm [22, 23]. We have additionally identified simple criteria, some of which are tied to specific data characteristics, that can be used by our framework in order to compare the behavior of learning algorithms in the presence of varying levels of noise [34].

[Data Series] There is an increasingly pressing need, by several applications in diverse domains (ranging from astronomy and biology, to electrical grids and manufacturing), for developing techniques able to index and mine very large collections of data series, in the order of hundreds of millions to billions. Evidently, this requirement calls for novel approaches and techniques for management and processing data series.

In this line of work, we have developed iSAX 2.0, a data structure designed for indexing and mining truly massive collections of data series [16], in collaboration with the University of California at Riverside. We showed that the main bottleneck in mining such massive datasets is the time taken to build the index, and we thus introduced the first bulk loading mechanism specifically tailored to a data series index, and reported the first published experiments to index one billion data series. Even though these results are promising for the practitioners, we observe that the analysis step cannot start before the lengthy indexing step ends. Removing this restriction is an interesting research direction.

In this area, we have also proposed fast and scalable techniques for pattern identification in data series streams [32]. The observation that in several cases the values in the data series are uncertain, has guided us to investigate this parameter of the problem. This value uncertainty may be due to the inherent imprecision of sensor observations, data aggregations, privacy-preserving transforms, or error-prone mining algorithms. Our study suggests that a fruitful research direction is to develop models for processing uncertain data series that take into account the temporal correlations in the data [18], which has not been considered so far.

5. ANALYTICS ON NON-STRUCTURED DATA

[Subjectivity Analysis] In the past years we have witnessed Sentiment Analysis and Opinion Mining becoming increasingly popular topics in Information Retrieval and Web data analysis, allowing us to capture sentiments and opinions, expressed in online user-generated content, at a large scale. Tracking how opinions or discussions evolve over time can help us identify interesting trends and patterns, and better understand the ways that information is propagated. The dbTrento group has been involved in research work relevant to the areas of Sentiment Analysis and Opinion Mining, and has spearheaded the work on Contradiction Analysis, which has also led to collaborations with HP Labs, the Qatar Computing Research Institute, and the Al Jazeera news broadcasting network. We have recently presented a comprehensive survey on the research problems in the above areas [56].

Our main focus has been the problem of finding sentiment-based contradictions at a large scale [57]. We defined two types of contradictions, depending on the distributions of opposite sentiments over time, namely, synchronous and asynchronous (sentiment-shift) contradictions, and introduced a novel measure of contradiction that accounts for the variability within and across data collections. We also proposed a scalable method for identifying both types of contradictions at different time scales that employs sentiment values on a continuous scale. An interesting direction of research is to characterize (e.g., in terms of demographics) and explain (e.g., in terms of news events) the identified contradictions, as well as generalize the proposed model to arbitrary opinion data (i.e., not just numeric sentiments) [55].

[Facet Discovery] Advances in social-media and user-generated content technologies have resulted in collecting extremely large volumes of user-annotated media; for instance photos (flickr), urls (del.icio.us), and others. All these platforms provide users with the capability of generating content and assigning *ad hoc* tags to this content. Motivated by applications in the domain of collaborative tagging, we have introduced the problem of diverse dimension decomposition, which can be used for facet discovery, where a dimension is a set of mutually exclusive tag-sets. The information theoretic mining framework we have proposed together with Yahoo! Research can be interpreted as a dimensionality-reducing transformation from the space of all tags to the space of orthogonal dimensions [53, 54].

6. FUTURE DIRECTIONS

The group continues the work on data and information management and analysis, with an emphasis on the problems arising from the scale of the data, their non-structured, heterogeneous and uncertain nature, and from specific application requirements, such as privacy guarantees on public administration data, or particular analysis tasks on scientific data. More specifically, the main research directions of the group are the following.

[Smart Cities] The availability of data and information on several different aspects of everyday life in digital format allows us to form a clear picture about the workings of a city, or a community in general. This can help us design tools for better managing fundamental principles relevant to citizens (such as privacy [17]) in new unexplored contexts, e.g., Big Open Data, and applications, e.g., Data Journalism. They will also enable us to react, follow on, predict, and influence various societal situations. In collaboration with the public administration and relevant industries, we will investigate novel techniques and methodologies that can help us achieve the above goals, even in new fields, like e-crime.

[User-Generated Content] Complementary to the previous direction is that of analyzing user-generated content. Given the wealth of such data on the web, we aim to develop a subjectivity analysis toolset that will take into account social structures and events information, and will offer intuitive analytics functionalities for understanding, explaining, and predicting trends and behaviors on the social web. Furthermore, the toolset will be predicting goals, user intentions and will be building dynamic user profiles from user generated content and user actions. Finally, it will be able to evaluate the quality of the provided information using the history of the users and the reactions of the crowds.

[Scientific Data] Through the ongoing collaboration with scientists, e.g., biologists and neuroscientists, who need to analyze large collections of data, usually on commodity hardware, we are aiming at providing them with tools that can efficiently perform complex analytics that take into account the special nature of their data and their intended tasks.

Acknowledgments

We thank all our postdoc, PhD and MSc students for their dedication and hard work: S. Bykau, A. Camerra, A. Chiasera, A. Cordioli, M. Dallachiesa, V. Falletta, G. Giannakopoulos, E. Iori, M. Lissandrini, A. Marascu, K. Mirylenka, D. Mottin, B. Nushi, D. Papadopoulou, M. Zerega, F. Rizzolo, C. Tsinaraki, M. Tsytsarau, and K. Zoumpatianos. We would also like to acknowledge the contributions of our internal and external collaborators, who made this research possible.

References

[1] B. Alexe, W. C. Tan, and Y. Velegrakis. Comparing and evaluating mapping systems with STBenchmark. *PVLDB*, 1(2), 2008.

[2] B. Alexe, W. C. Tan, and Y. Velegrakis. STBenchmark: towards a benchmark for mapping systems. *PVLDB*, 1(1), 2008.

[3] N. Augsten, D. Barbosa, M. H. Böhlen, and T. Palpanas. Tasm: Top-k approximate subtree matching. In *ICDE*, 2010.

[4] N. Augsten, D. Barbosa, M. H. Böhlen, and T. Palpanas. Efficient top-k approximate subtree matching in small memory. *TKDE*, 23(8), 2011.

[5] B. Bazzanella, T. Palpanas, and H. Stoermer. Towards a general entity representation model. In *IRI*, 2009.

[6] S. Bergamaschi, E. Domnori, F. Guerra, R. Trillo Lado, and Y. Velegrakis. Keyword Search over Relational Databases: A Metadata Approach. In *SIGMOD*, 2011.

[7] S. Bergamaschi, E. Domnori, F. Guerra, M. Orsini, R. T. Lado, and Y. Velegrakis. Keymantic: Semantic Keyword based Searching in Data Integration Systems. *PVLDB*, 3(2), 2010.

[8] S. Bergamaschi, F. Guerra, S. Rota, and Y. Velegrakis. A Hidden Markov Model Approach to Keyword-based Search over Relational Databases. In *ER*, 2011.

[9] S. Bergamaschi, F. Guerra, S. Rota, and Y. Velegrakis. KEYRY: a Keyword-based Search Engine over Relational Databases based on a Hidden Markov Model. In *ER*, 2011.

[10] S. Bergamaschi, F. Guerra, S. Rota, and Y. Velegrakis. Understanding Linked Open Data through Keyword Searching: the KEYRY approach. In *LWDM*, 2011.

[11] A. Bonifati and Y. Velegrakis. Schema Matching and Mapping: From Usage to Evaluation. In *EDBT*, 2011.

[12] P. Bouquet, T. Palpanas, H. Stoermer, and M. Vignolo. A conceptual model for a web-scale entity name system. In *ASWC*, 2009.

[13] S. Bykau, N. Kiyavitskaya, C. Tsinaraki, and Y. Velegrakis. Bridging the Gap Between Heterogeneous and Semantically Diverse Content of Different Disciplines. In *FLEXDBIST*, 2010.

[14] S. Bykau, J. Mylopoulos, F. Rizzolo, and Y. Velegrakis. Supporting Queries Spanning Across Phases of Evolving Artifacts using Steiner Forests. In *CIKM*, 2011.

[15] S. Bykau, J. Mylopoulos, F. Rizzolo, and Y. Velegrakis. On Modeling and Querying Concept Evolution. *Journal on Data Semantics*, 1, 2012.

[16] A. Camerra, T. Palpanas, J. Shieh, and E. J. Keogh. isax 2.0: Indexing and mining one billion time series. In *ICDM*, 2010.

[17] A. Chiasera, F. Casati, F. Daniel, and Y. Velegrakis. Engineering Privacy Requirements in Business Intelligence Applications. In *SDM*, 2008.

[18] M. Dallachiesa, B. Nushi, K. Mirylenka, and T. Palpanas. Uncertain time-series similarity: Return to the basics. *PVLDB*, 5(11), 2012.

[19] R. De Virgilio, F. Guerra, and Y. Velegrakis. *Semantic Search over the Web*. Springer, 2012.

[20] R. Fagin, L. M. Haas, M. A. Hernández, R. J. Miller, L. Popa, and Y. Velegrakis. Clio: Schema mapping creation and data exchange. In A. Borgida, V. K. Chaudhri,

P. Giorgini, and E. S. K. Yu, editors, *Conceptual Modeling: Foundations and Applications*, volume 5600 of *Lecture Notes in Computer Science*. Springer, 2009.

[21] G. Giannakopoulos and T. Palpanas. Content and type as orthogonal modeling features: a study on user interest awareness in entity subscription services. *International Journal of Advances on Networks and Services*, 3(2), 2010.

[22] G. Giannakopoulos and T. Palpanas. The effect of history on modeling systems' performance: The problem of the demanding lord. In *ICDM*, 2010.

[23] G. Giannakopoulos and T. Palpanas. Revisiting the effect of history on learning performance: The problem of the demanding lord. *KAIS*, accepted for publication.

[24] O. Hassanzadeh, A. Kementsietsidis, and Y. Velegrakis. Data Management Issues on the Semantic Web. In *ICDE*, 2012.

[25] E. Ioannou, W. Nejdl, C. Niederee, and Y. Velegrakis. OntheFly Entity-Aware Query Processing in the Presence of Linkage. *PVLDB*, 3(1), 2010.

[26] E. Ioannou, W. Nejdl, C. Niederee, and Y. Velegrakis. LinkDB: A Probabilistic Linkage Database System. In *SIGMOD*, 2011.

[27] E. Ioannou, C. Niederee, and Y. Velegrakis. Enabling Entity-Based Aggregators for Web 2.0 data. In *WWW*, 2010.

[28] E. Iori, A. Simitsis, T. Palpanas, K. Wilkinson, and S. Harizopoulos. Cloudalloc: A monitoring and reservation system for compute clusters. In *SIGMOD*, 2012.

[29] A. Katifori, C. Nikolaou, M. Platakis, Y. Ioannidis, A. Tympas, M. Koubarakis, N. Sarris, V. Tountopoulos, E. Tzoanos, S. Bykau, N. Kiyavitskaya, C. Tsinaraki, and Y. Velegrakis. The Papyrus Digital Library: Discovering History in the News. In *TPDL*, 2011.

[30] Y. Kotidis, D. Srivastava, and Y. Velegrakis. Updates Through Views: A New Hope. In *ICDE*, pages 13–24, 2006.

[31] N. Manerikar and T. Palpanas. Frequent items in streaming data: An experimental evaluation of the state-of-the-art. *DKE*, 68(4), 2009.

[32] A. Marascu, S. A. Khan, and T. Palpanas. Scalable similarity matching in streaming time series. In *PAKDD*, 2012.

[33] Z. Miklos, N. Bonvin, P. Bouquet, M. Catasta, D. Cordioli, P. Fankhauser, J. Gaugaz, E. Ioannou, H. Koshutanski, A. Mana, C. Niederee, T. Palpanas, and H. Stoermer. From web data to entities and back. In *CAiSE*, 2010.

[34] K. Mirylenka, G. Giannakopoulos, and T. Palpanas. Srf: A framework for the study of classifier behavior under training set mislabeling noise. In *PAKDD*, 2012.

[35] D. Mottin, T. Palpanas, and Y. Velegrakis. Entity Ranking Using Click-Log Information. *Intelligent Data Analysis Journal*, 17:5, 2013.

[36] I. Ntoutsi, A. Zimek, T. Palpanas, P. Kröger, and H.-P. Kriegel. Density-based projected clustering over high dimensional data streams. In *SDM*, 2012.

[37] T. Palpanas. Real-time data analytics in sensor networks. In C. Aggarwal, editor, *Managing and Mining Sensor Data*. Springer, 2012.

[38] T. Palpanas, M. Vlachos, E. J. Keogh, and D. Gunopulos. Streaming time series summarization using user-defined amnesic functions. *TKDE*, 20(7), 2008.

[39] G. Papadakis, G. Giannakopoulos, C. Niederée, T. Palpanas, and W. Nejdl. Detecting and exploiting stability in evolving heterogeneous information spaces. In *JCDL*, 2011.

[40] G. Papadakis, E. Ioannou, C. Niederée, T. Palpanas, and W. Nejdl. Beyond 100 million entities: large-scale blocking-based resolution for heterogeneous data. In *WSDM*, 2012.

[41] G. Papadakis, E. Ioannou, T. Palpanas, C. Niederee, and W. Nejdl. A blocking framework for entity resolution in highly heterogeneous information spaces. *TKDE*, accepted for publication.

[42] A. Presa, Y. Velegrakis, F. Rizzolo, and S. Bykau. Modeling Associations through Intensional Attributes. In *ER*, 2009.

[43] U. Raza, A. Camerra, A. L. Murphy, T. Palpanas, and G. P. Picco. What does model-driven data acquisition really achieve in wireless sensor networks? In *PerCom*, Lugano, Switzerland, 2012.

[44] F. Rizzolo, Y. Velegrakis, J. Mylopoulos, and S. Bykau. Modeling Concept Evolution: A Historical Perspective. In *ER*, 2009.

[45] D. Srivastava and Y. Velegrakis. Intensional Associations between Data and Metadata. In *SIGMOD*, pages 401–412, 2007.

[46] D. Srivastava and Y. Velegrakis. MMS: Using Queries As Data Values for Metadata Management. In *ICDE*, pages 1481–1482, 2007.

[47] D. Srivastava and Y. Velegrakis. Using Queries to Associate Metadata with Data. In *ICDE*, pages 1451–1453, 2007.

[48] I. Stanoi, G. A. Mihaila, T. Palpanas, and C. A. Lang. Whitewater: Distributed processing of fast streams. *TKDE*, 19(9), 2007.

[49] S. Subramaniam, V. Kalogeraki, and T. Palpanas. Distributed Real-Time Detection and Tracking of Homogeneous Regions in Sensor Networks. In *RTSS*, Rio de Janeiro, Brazil, 2006.

[50] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online outlier detection in sensor data using non-parametric models. In *VLDB*, 2006.

[51] F. I. Tantono, N. Manerikar, and T. Palpanas. Efficiently discovering recent frequent items in data streams. In *SSDBM*, 2008.

[52] C. Tsinaraki, Y. Velegrakis, N., and J. Mylopoulos. A Context-based Model for the Interpretation of Polysemous Terms. In *ODBASE*, 2010.

[53] M. Tsytsarau, F. Bonchi, A. Gionis, and T. Palpanas. Diverse dimension decomposition of an itemset space. In *ICDM*, 2011.

[54] M. Tsytsarau, F. Bonchi, A. Gionis, and T. Palpanas. Diverse dimension decomposition for itemset spaces. *KAIS*, accepted for publication.

[55] M. Tsytsarau and T. Palpanas. Towards a framework for detecting and managing opinion contradictions. In *ICDM Workshops*, 2011.

[56] M. Tsytsarau and T. Palpanas. Survey on mining subjective data on the web. *DMKD*, 24(3), 2012.

[57] M. Tsytsarau, T. Palpanas, and K. Denecke. Scalable discovery of contradictions on the web. In *WWW*, 2010.

Temporal features in SQL:2011

Krishna Kulkarni, Jan-Eike Michels (IBM Corporation)
{krishnak, janeike}@us.ibm.com

ABSTRACT

SQL:2011 was published in December of 2011, replacing SQL:2008 as the most recent revision of the SQL standard. This paper covers the most important new functionality that is part of SQL:2011: the ability to create and manipulate temporal tables.

1. Introduction

SQL is the predominant database query language standard published jointly by ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission). In December 2011, ISO/IEC published the latest edition of the SQL standard, SQL:2011. A recent article in *SIGMOD Record* provides a brief survey of the new features in SQL:2011 [1]. Because of space constraints, it did not cover the most important new feature in SQL:2011: the ability to create and manipulate temporal tables, i.e., tables whose rows are associated with one or more temporal periods. This is the subject of the current article.

2. Temporal data support

Extensions to support temporal data[1] in SQL have long been desired. There is a large body of research papers, conference publications, and books on this topic, some dating back to the early 1980s. For more details, we refer the readers to an extensive (but outdated) bibliography [2] and to books such as [3] and [4].

Though the previous academic research produced a large number of solutions, the commercial adoption has been rather slow. It is only recently that commercial database management systems (DBMSs) have begun to offer SQL extensions for managing temporal data [5, 6, 7]. Prior to this development, users were forced to implement temporal support as part of the application logic, which often resulted in expensive development cycles and complex, hard-to-maintain code.

In 1995, the ISO SQL committee initiated a project to create a new part of SQL standard devoted to the language extensions for the temporal data support. A set of language extensions based on (but not identical to) TSQL2 [8] were submitted for standardization at that time. Unfortunately, these proposals generated considerable controversy (see [9] for more details), and failed to get adequate support from the ISO SQL committee's membership. In addition, there was no indication that key DBMS vendors were planning to implement these extensions in their products. Eventually, the work on this new part was cancelled in 2001.

Recently, a new set of language extensions for temporal data support were submitted to and accepted by the ISO SQL committee. These language extensions are now part of SQL:2011 Part 2, SQL/Foundation [10], instead of appearing as a new part. There is currently at least one commercial implementation [5] based on these extensions that the authors are aware of.

2.1 Periods

The cornerstone of temporal data support in SQL:2011 is the ability to define and associate time periods with the rows of a table. Essentially, a time period is a mathematical interval on the timeline, demarcated by a start time and an end time.

Many treatments of temporal databases introduce a period data type, defined as an ordered pair of two datetime values, for the purpose of associating time periods with the rows of a table. SQL:2011 has not taken this route. Adding a new data type to the SQL standard (or to an SQL product) is a costly venture because of the need to support the new data type in the tools and other software packages that form the ecosystem surrounding SQL. For example, if a period type were added to SQL, then it would have to also be added to the stored procedure language, to all database APIs such as JDBC, ODBC, and .NET, as well as to the surrounding technology such as ETL products, replication solutions, and others. There must also be some means of communicating period values to host languages that do not support period as a native data type, such as C or Java. These factors can potentially slow down the adoption of new type for a long time.

1. Note that there is no single, commonly accepted definition of the term "temporal data". For the purposes of this article, we define "temporal data" to mean any data with one or more associated time periods during which that data is deemed to be effective or valid along some time dimension.

Instead of adding a period type, SQL:2011 adds *period definitions* as metadata to tables. A period definition is a named table component, identifying a pair of columns that capture the period start and the period end time. `CREATE TABLE` and `ALTER TABLE` statements are enhanced with syntax to create or destroy period definitions. The period start and end columns are conventional columns, with separate names. The period name occupies the same name space as column names, i.e., a period cannot have the same name as a column.

SQL:2011 has adopted a closed-open period model, i.e., a period represents all times starting from and including the start time, continuing to but excluding the end time. For a given row, the period end time must be greater than its period start time; in fact, declaring a period definition in a table implies a table constraint that enforces this property.

The literature on temporal databases recognizes two dimensions of time for temporal data support, e.g., see [3]:

- *valid time*, the time period during which a row is regarded as correctly reflecting reality by the user of the database.
- *transaction time*, the time period during which a row is committed to or recorded in the database.

For any given row, its transaction time may arbitrarily differ from its valid time. For example, in an insurance database, information about a policy may get inserted much before that policy comes into effect.

In SQL:2011, transaction time support is provided by system-versioned tables, which in turn contain the *system-time period*, and valid time support is provided by tables containing an *application-time period*[2]. The name of the system-time period is specified by the standard as `SYSTEM_TIME`.The name of an application-time period can be any user-defined name. Users are allowed to define at most one application-time period and at most one system-time period per table.

One of the advantages of the SQL:2011 approach over an approach based on the period data type is that it allows existing databases that capture period information in a pair of datetime columns to take advantage of the SQL:2011 extensions more easily. Ever since DBMSs have been on the scene, users have been building their own solutions for handling temporal data as part of their application logic. Since most DBMSs do not support a period type, applications dealing with temporal data have tended to capture the period information using a pair of columns of datetime data type. It would be very expensive for users invested in such solutions to replace them with a solution that uses a single column of period type.

2. Interestingly, SQL:2011 manages to provide this support without actually defining or using the terms “temporal data” or “temporal table”.

2.2 Application-time period tables

Application-time period tables are intended for meeting the requirements of applications that are interested in capturing time periods during which the data is believed to be valid in the real world. A typical example of such applications is an insurance application, where it is necessary to keep track of the specific policy details of a given customer that are in effect at any given point in time.

A primary requirement of such applications is that the user be put in charge of setting the start and end times of the validity period of rows, and the user be free to assign any time values, either in the past, current or in the future, for the start and end times. Another requirement of such applications is that the user be permitted to update the validity periods of the rows as errors are discovered or new information is made available.

Any table that contains a period definition with a user-defined name is an application-time period table. For example:

```
CREATE TABLE Emp(
ENo INTEGER,
EStart DATE,
EEnd DATE,
EDept INTEGER,
PERIOD FOR EPeriod (EStart, EEnd)
)
```

Users can pick any name they want for the name of the period as well as for the names of columns that act as the start and end columns of the period. The data types of the period start and end columns must be either DATE or a timestamp type, and data types of both columns must be the same.

The conventional INSERT statement provides sufficient support for setting the initial values of application-time period start and end columns. For example, the following INSERT statement inserts one row into the `Emp` table:

```
INSERT INTO Emp
VALUES (22217,
         DATE ‘2010-01-01’,
         DATE '2011-11-12', 3)
```

The resulting table looks as shown below (assuming it was empty before):

Eno	EStart	EEnd	EDept
22217	2010-01-01	2011-11-12	3

The conventional UPDATE statement can be used to modify the rows of application-time period tables (including the application-time period start and end times). Similarly, the conventional DELETE statement can be used to delete rows of application-time period tables.

A new feature in SQL:2011 is the ability to specify changes that are effective within a specified period. This is provided by a syntactic extension to both UPDATE and DELETE statements that lets users specify the period of interest. For example, the following UPDATE statement changes the department of the employee whose number is 22217 to 4 for the period from Feb. 3, 2011 to Sept. 10, 2011:

```
UPDATE Emp
  FOR PORTION OF EPeriod
    FROM DATE '2011-02-03'
    TO DATE '2011-09-10'
SET EDept = 4
WHERE ENo = 22217
```

To execute this statement, the DBMS locates all rows whose application-time period overlaps the period *P* from Feb. 3, 2011 to Sept. 10, 2011. Recall that periods follow closed-open semantics in SQL:2011, so *P* includes Feb. 3, 2011 but excludes Sept. 10, 2011. Any overlapping row whose application-time period is contained in *P* is simply updated. If an overlapping row whose application-time period has a portion either strictly before or strictly after *P*, then that row gets split into two or three contiguous rows depending on the extent of overlap, and of these, the row whose application-time period is contained in *P* is updated. For example, suppose the following is the only overlapping row:

ENo	EStart	EEnd	EDept
22217	2010-01-01	2011-11-12	3

Note that the application-time period of the above row extends beyond *P* at both ends. The result of the UPDATE statement will be these three rows:

ENo	EStart	EEnd	EDept
22217	2010-01-01	2011-02-03	3
22217	2011-02-03	2011-09-10	4
22217	2011-09-10	2011-11-12	3

In this example, the row whose EDept value is updated to 4 is regarded as the original row and hence, UPDATE triggers fire for this row. The other two rows are regarded as newly inserted rows, so INSERT triggers fire for them.

The DELETE statement is similarly enhanced with FOR PORTION OF syntax to facilitate deletes that are only effective within a specified period. For example, the following DELETE statement removes the employee whose number is 22217 for the period from Feb. 3, 2011 to Sept. 10, 2011:

```
DELETE Emp
  FOR PORTION OF EPeriod
    FROM DATE '2011-02-03'
    TO DATE '2011-09-10'
WHERE ENo = 22217
```

Similar to the UPDATE example, any row whose application-time period is contained in *P* from Feb. 3, 2011 to Sept. 10, 2011 is simply deleted. If an overlapping row whose application-time period has a portion either strictly before or strictly after *P*, then that row gets split into two or three contiguous rows, and of these, the row whose application-time period is contained in *P* is deleted. For example, suppose the following is the only overlapping row:

ENo	EStart	EEnd	EDept
22217	2010-01-01	2011-11-12	3

The result of the statement will be these two rows:

ENo	EStart	EEnd	EDept
22217	2010-01-01	2011-02-03	3
22217	2011-09-10	2011-11-12	3

In this example, the result is the deletion of the original row and the insertion of two new rows; DELETE triggers fire for the deleted row and INSERT triggers fire for the newly inserted rows.

2.2.1 Primary keys on application-time period tables

The last section gave an example of an Emp table in which one might expect that ENo is the primary key. However, looking at the sample result of the UPDATE statement, there are three rows all with ENo 22217. This example shows that the primary key must also include the application-time period columns EStart and EEnd.

Simply adding EStart and EEnd to the primary key will not be sufficient though. Consider the following data:

ENo	EStart	EEnd	EDept
22217	2010-01-01	2011-09-10	3
22217	2010-02-03	2011-11-12	4

The triples (22217, 2010-01-01, 2011-09-10) and (22217, 2010-02-03, 2011-11-12) are not duplicates so they would be acceptable values for a conventional primary key on these three columns. But note that the application-time periods of these rows overlap. Semantically, this says that the employee with ENo 22217 belongs to two departments, 3 and 4, during the period from Feb. 3, 2010 through Sept. 10, 2011. Perhaps the user wishes to allow an employee to belong to two departments; however, the more typical requirement is that an employee belongs to exactly one department at any given time. To achieve that, it must be possible to forbid overlapping application-time periods, which can be specified with this syntax:

```
ALTER TABLE Emp
ADD PRIMARY KEY (ENo,
 EPeriod WITHOUT OVERLAPS)
```

With this primary key definition, the sample data is prohibited as a constraint violation.

2.2.2 Referential constraints on application-time period tables

Continuing the preceding example, suppose there is another table with the following definition:

```
CREATE TABLE Dept(
DNo INTEGER,
DStart DATE,
DEnd DATE,
DName VARCHAR(30),
PERIOD FOR DPeriod (DStart, DEnd),
PRIMARY KEY (DNo,
  DPeriod WITHOUT OVERLAPS)
)
```

Assume also that we want to make sure that at every point in time, every value in EDept column corresponds to some value of DNo column in Dept table, i.e., every employee at every point in time during her employment belongs to a department that actually exists at that point in time. How should this work? Let's look at some sample data. Assume the Emp table contains the following rows:

ENo	EStart	EEnd	EDept
22218	2010-01-01	2011-02-03	3
22218	2011-02-03	2011-11-12	4

Assume the Dept table contains the following rows:

DNo	DStart	DEnd	DName
3	2009-01-01	2011-12-31	Test
4	2011-06-01	2011-12-31	QA

Looking strictly at the values of EDept column of the Emp table and the DNo column of the Dept table, we may conclude that the conventional referential integrity constraint involving the two tables is satisfied. But note that the employee with ENo 22218 is assigned to the department with DNo 4 from Feb. 3, 2011 to Nov. 12, 2011, but there is no department with DNo 4 for the period from Feb. 3, 2011 to June 1, 2011. Clearly, this violates our requirement that every value of EDept column in Emp table corresponds to some value of DNo column in Dept table at every point in time. To disallow such a situation, it must be possible to forbid a row in a child table whose application-time period is not contained in the application-time period of a matching row in the parent table, which can be specified with this syntax:

```
ALTER TABLE Emp
ADD FOREIGN KEY
   (Edept, PERIOD EPeriod)
   REFERENCES Dept
   (DNo, PERIOD DPeriod)
```

With this referential constraint definition, the sample data is prohibited as a constraint violation.

More generally, for a given child row, it is not necessary that there exists exactly one matching row in the parent table whose application-time period contains the application-time period of the child row. As long as the application-time period of a row in the child table is contained in the union of application-time periods of two or more contiguous matching rows in the parent table, the referential constraint is considered satisfied.

2.2.3 Querying application-time period tables

In SQL:2011, application-time period tables can be queried using the regular query syntax. For example, to

retrieve the department where the employee 22217 worked as of January 2, 2011, one can express the query as:

```
SELECT Name, Edept
FROM Emp
WHERE ENo = 22217
  AND EStart <= DATE '2011-01-02'
  AND EEnd > DATE '2011-01-02'
```

A simpler way to formulate the above query would be to employ one of the period predicates provided in SQL:2011 for expressing conditions involving periods: CONTAINS, OVERLAPS, EQUALS, PRECEDES, SUCCEEDS, IMMEDIATELY PRECEDES, and IMMEDIATELY SUCCEEDS. For example, the above query could also be expressed using the CONTAINS predicate, as shown below:

```
SELECT Ename, Edept
FROM Emp
WHERE ENo = 22217 AND
  EPeriod CONTAINS DATE '2011-01-02'
```

If one wanted to know all the departments where the employee whose number is 22217 worked during the period from January 1, 2010 to January 1, 2011, one could formulate the query as:

```
SELECT Ename, Edept
FROM Emp
WHERE ENo = 22217
  AND EStart < DATE '2011-01-01'
  AND EEnd > DATE '2010-01-01'
```

Note that the period specified in the above query uses the closed-open model, i.e., the period includes January 1, 2010 but excludes January 1, 2011. Alternatively, the same query could be expressed using the OVERLAPS predicate as:

```
SELECT Ename, Edept
FROM Emp
WHERE ENo = 22217 AND
  EPeriod OVERLAPS
     PERIOD (DATE '2010-01-01',
             DATE '2011-01-01')
```

Period predicates are functionally similar to (but not identical to) the well-known Allen's interval operators [11]. The correspondence between SQL's period predicates and Allen's operators is as follows:

- The predicate "`X OVERLAPS Y`" in SQL:2011 is equivalent to the Boolean expression using Allen's operators "`(X overlaps Y) OR (X overlapped_by Y) OR (X during Y) OR (X contains Y) OR (X starts Y) OR (X started_by Y) OR (X finishes Y) OR (X finished_by Y) OR (X equal Y)`". Note that Allen's `overlaps` operator is not a true test of period overlap. Intuitively, two periods are considered overlapping if they have at least one time point in common. This is not true for Allen's `overlaps` operator. In contrast, SQL:2011's `OVERLAPS` predicate is a true test of period overlap. Also, SQL:2011's `OVERLAPS` predicate is symmetric, i.e, if "`X OVERLAPS Y`" is true, then "`Y OVERLAPS X`" is also true. This is again not true for Allen's `overlaps` operator.
- The predicate "`X CONTAINS Y`" in SQL:2011 is equivalent to the Boolean expression using Allen's operators "`(X contains Y) OR (X starts Y) OR (X finishes Y) OR (X equal Y)`". Note that Allen's `contains` operator is not a true test of period containment. Intuitively, period `X` is considered containing period `Y` if every time point in `Y` is also in `X`. This is not true for Allen's `contains` operator. In contrast, SQL:2011's `CONTAINS` predicate is a true test of period containment.
- The predicate "`X PRECEDES Y`" in SQL:2011 is equivalent to the Boolean expression using Allen's operators "`(X before Y) OR (X meets Y)`".
- The predicate "`X SUCCEEDS Y`" in SQL:2011 is equivalent to the Boolean expression using Allen's operators "`(X after Y) OR (X met_by Y)`".
- The predicates "`X EQUALS Y`", "`X IMMEDIATELY PRECEDES Y`", and "`X IMMEDIATELY SUCCEEDS Y`" in SQL:2011 are equivalent to the Allen's operators "`X equal Y`", "`X meets Y`", and "`X met_by Y`", respectively.

2.3 System-versioned tables

System-versioned tables are intended for meeting the requirements of applications that must maintain an accurate history of data changes either for business reasons, legal reasons, or both. A typical example of such applications is a banking application, where it is necessary to keep previous states of customer account information so that customers can be provided with a detailed history of their accounts. There are also plenty of examples where certain institutions are required by law to preserve historical data for a specified length of time to meet regulatory and compliance requirements.

A key requirement of such applications is that any update or delete of a row must automatically preserve the old state of the row before performing the update or delete. Another important requirement is that the system, rather than the user, maintains the start and end times of the periods of the rows, and that users be unable to modify the content of historical rows or the periods associated with any of the rows. Any updates to the periods of rows in a system-versioned table must be performed only by the system as a result of updates to the non-period columns of the table or as a result of row deletions. This provides the guarantee that the recorded

history of data changes cannot be tampered with, which is critical to meet auditing and compliance regulations.

Any table that contains a period definition with the standard-specified name, SYSTEM_TIME, and includes the keywords WITH SYSTEM VERSIONING in its definition is a system-versioned table. Similar to application-time period tables, users can pick any name they want for the names of columns that act as the start and end columns of the SYSTEM_TIME period. Though SQL:2011 allows the data types of the period start and end columns to be either DATE or a timestamp type (as long as the data types of both columns are the same), in practice, most implementations will provide the TIMESTAMP type with the highest fractional seconds precision as the data type for the system-time period start and end columns. For example:

```
CREATE TABLE Emp
 ENo INTEGER,
 Sys_start TIMESTAMP(12) GENERATED
  ALWAYS AS ROW START,
 Sys_end TIMESTAMP(12) GENERATED
  ALWAYS AS ROW END,
 EName VARCHAR(30),
 PERIOD FOR SYSTEM_TIME (Sys_start,
  Sys_end)
) WITH SYSTEM VERSIONING
```

Similar to application-time periods, system-time periods use closed-open period model. At any given point in time, a row in a system-versioned table is regarded as *current system row* if the system-time period of that row contains the current time. A row that is not a current system row is regarded as a *historical system row.*

System-versioned tables differ from application-time period tables in the following respects:

1) In contrast to the application-time period tables, users are not allowed to assign or change the values of Sys_start or Sys_end columns; they are assigned (and changed) automatically by the database system. This is the reason why the definitions of Sys_start or Sys_end columns must include the keywords GENERATED ALWAYS.
2) INSERT into a system-versioned table automatically sets the value of Sys_start column to the *transaction timestamp*, a special value associated with every transaction[3], and sets the value of Sys_end column to the highest value of the column's data type. For example, assume that the following INSERT statement executed in a transaction whose transaction timestamp is 2012-01-01 09:00:00[4]:

```
INSERT INTO Emp (ENo, EName)
   VALUES (22217, 'Joe')
```

The resulting table looks as shown below (assuming it was empty before):

ENo	Sys_Start	Sys_End	EName
22217	2012-01-01 09:00:00	9999-12-31 23:59:59	Joe

3) UPDATE and DELETE on system-versioned tables only operate on current system rows. Users are not allowed to update or delete historical system rows. Users are also not allowed to modify the system-time period start or the end time of both current system rows and historical system rows.
4) UPDATE and DELETE on system-versioned tables result in the automatic insertion of a historical system row for every current system row that is updated or deleted.

An UPDATE statement on a system-versioned table first inserts a copy of the old row with its system-time period end time set to the transaction timestamp, indicating that the row ceased to be current as of the transaction timestamp. It then updates the row while changing its system-period start time to the transaction timestamp, indicating that the updated row to be the current system row as of the transaction timestamp. For example, suppose the current system row with ENo 22217 is as shown below:

ENo	Sys_Start	Sys_End	EName
22217	2012-01-01 09:00:00	9999-12-31 23:59:59	Joe

The following UPDATE statement changes the name of the employee whose number is 22217 from Joe to Tom effective from the transaction timestamp of the transaction in which the UPDATE statement was executed:

```
UPDATE Emp
SET EName = 'Tom'
WHERE ENo = 22217
```

A historical system row that corresponds to the state of the row prior to the update is first inserted and then

3. SQL:2011 leaves it up to SQL-implementations to pick an appropriate value for the transaction timestamp of a transaction, but it does require the transaction timestamp of a transaction to remain fixed during the entire transaction.

4. Note that we are not showing the fractional part of seconds in any of the examples in this Section.

the update is performed. Assuming the above statement is executed in a transaction with the transaction timestamp 2012-02-03 10:00:00, the final result will be these two rows:

ENo	Sys_Start	Sys_End	EName
22217	2012-01-01 09:00:00	2012-02-03 10:00:00	Joe
22217	2012-02-03 10:00:00	9999-12-31 23:59:59	Tom

In this example, the row whose name is Tom is the updated row; `UPDATE` triggers fire for this row. Note that the insertion of historical system rows does not fire any `INSERT` triggers for the inserted rows. Note also that historical system rows created as a result of sequence of updates for a given row form one contiguous chain without any gap between their system-time periods.

A `DELETE` statement on a system-versioned table does not actually delete the qualifying rows; instead it changes the system-time period end time of those row to the transaction timestamp, indicating that those rows ceased to be current as of the transaction timestamp. For example, suppose that the current system row with ENo 22217 is as shown below:

ENo	Sys_Start	Sys_End	EName
22217	2012-01-01 09:00:00	9999-12-31 23:59:59	Joe

The following `DELETE` statement simply changes the system-time period end time of the current system row for the employee 22217 to the transaction timestamp of the transaction in which the `DELETE` statement was executed:

```
DELETE FROM Emp
WHERE ENo = 22217
```

Assuming the above statement is executed in a transaction with the transaction timestamp 2012-06-01 00:00:00, the final result will be the following row:

ENo	EStart	EEnd	EName
22217	2012-01-01 09:00:00	2012-06-01 00:00:00	Joe

In this example, `DELETE` triggers fire for the row selected for deletion.

Note that in contrast to the application-time period tables, `FOR PORTION OF SYSTEM_TIME` is not needed (and hence not allowed) for the `UPDATE` and `DELETE` statements on system-versioned tables.

2.3.1 Primary key and referential constraints on system-versioned tables

The definition and enforcement of constraints on system-versioned tables is considerably simpler than the definition and enforcement of constraints on application-time period tables. This is because constraints on system-versioned tables need only be enforced on the current system rows. Historical system rows in a system-versioned table form immutable snapshots of the past. Any constraints that were in effect when a historical system row was created would have already been checked when that row was a current system row, so there is never any need to enforce constraints on historical system rows. Consequently, there is no need to include the system-period start and end columns or the period name in the definition of primary key and referential constraints on system-versioned tables. For example, the following ALTER TABLE statement specifies `ENo` column as the primary key of `Emp` table:

```
ALTER TABLE Emp
ADD PRIMARY KEY (ENo)
```

The above constraint ensures there exists exactly one current system row with a given `ENo` value.

Similarly, the following ALTER TABLE statement specifies a referential constraint between `Emp` and `Dept` tables:

```
ALTER TABLE Emp
ADD FOREIGN KEY (Edept)
   REFERENCES Dept (DNo)
```

The above constraint is again enforced only on the current system rows of `Emp` and `Dept` tables.

2.3.2 Querying system-versioned tables

Because system-versioned tables are intended primarily for tracking historical data changes, queries on system-versioned tables often tend to be concerned with retrieving the table content as of a given point in time or between any two given points in time. SQL:2011 provides three syntactic extensions for this specific purpose. These are allowed only in queries on system-versioned tables.

The first extension is the `FOR SYSTEM_TIME AS OF` syntax that is useful for querying the table content as of a specified point in time. For example, the following query retrieves the rows of `Emp` that were current as of Jan. 2, 2011:

```
SELECT ENo,EName,Sys_Start,Sys_End
FROM Emp FOR SYSTEM_TIME AS OF
   TIMESTAMP '2011-01-02 00:00:00'
```

The above query returns all rows whose system-time period start time is less than or equal to the specified timestamp and whose system-time period end time is greater than the specified timestamp.

The second and third extensions allow for retrieving the content of a system-versioned table between any two points in time. The following query returns all rows that were current starting from `TIMESTAMP '2011-01-02 00:00:00'` up to (but not including) `TIMESTAMP '2011-12-31 00:00:00'`:

```
SELECT ENo,EName,Sys_Start,Sys_End
FROM Emp FOR SYSTEM_TIME FROM
  TIMESTAMP '2011-01-02 00:00:00'TO
  TIMESTAMP '2011-12-31 00:00:00'
```

In contrast, the following query returns all rows that were current starting from `TIMESTAMP '2011-01-02 00:00:00'` up to (and including) `TIMESTAMP '2011-12-31 00:00:00'`:

```
SELECT ENo,EName,Sys_Start,Sys_End
FROM Emp FOR SYSTEM_TIME BETWEEN
  TIMESTAMP '2011-01-02 00:00:00'AND
  TIMESTAMP '2011-12-31 00:00:00'
```

Note that the period specified in the (`FROM ... TO ...`) corresponds to a closed-open period model while the period specified in the (`BETWEEN ... AND ...`) corresponds to a closed-closed period model.

If a query on system-versioned tables does not specify any of the above three syntactic options, then that query is assumed to specify `FOR SYSTEM_TIME AS OF CURRENT_TIMESTAMP` by default and the query returns only the current system rows as the result. For example, the following query returns only the current system rows of `Emp` table:

```
SELECT ENo,EName,Sys_Start,Sys_End
FROM Emp
```

The choice of returning current systems rows as the default is especially suited for those applications where retrieval of current system rows is the most frequent operation. In addition, it also helps with the database migration in that applications running on non-system-versioned tables would continue to work and produce the same results when those tables are converted to system-versioned tables.

Finally, to retrieve both current and historical system rows of a system-versioned table, one can use a query of the kind shown below:

```
SELECT ENo,EName,Sys_Start,Sys_End
FROM Emp FOR SYSTEM_TIME FROM
 TIMESTAMP '0001-01-01 00:00:00' TO
 TIMESTAMP '9999-12-31 23:59:59'
```

2.4 Bitemporal tables

A table may be both a system-versioned table and an application-time period table[5]. For example:

```
CREATE TABLE Emp(
  ENo INTEGER,
  EStart DATE,
  EEnd DATE,
  EDept INTEGER,
  PERIOD FOR EPeriod (EStart, EEnd),
  Sys_start TIMESTAMP(12) GENERATED
      ALWAYS AS ROW START,
  Sys_end TIMESTAMP(12) GENERATED
      ALWAYS AS ROW END,
  EName VARCHAR(30),
  PERIOD FOR SYSTEM_TIME
      (Sys_start, Sys_end),
  PRIMARY KEY (ENo,
      EPeriod WITHOUT OVERLAPS),
  FOREIGN KEY
      (Edept, PERIOD EPeriod)
   REFERENCES Dept
      (DNo, PERIOD DPeriod)
) WITH SYSTEM VERSIONING
```

Rows in such tables are associated with both the system-time period and the application-time period. Such tables are very useful for capturing both the periods during which facts were believed to be true in the real world as well as the periods during which those facts were recorded in the database. For example, while employed, an employee may change names. Typically the name changes legally at a specific time (for example, a marriage) but the name is not changed in the database concurrently with the legal change. In that case, the system-time period automatically records when a particular name is known to the database, and the application-time period records when the name was legally effective. Successive updates to bitemporal tables can journal complex twists and turns in the state of knowledge captured by the database.

Bitemporal tables combine the capabilities of both system-versioned and application-time period tables. As in the case of application-time period tables, the user is in charge of supplying values for the application-time period start and end columns. As in the case of system-versioned tables, `INSERT` into such a table automatically sets the value of system-time period start column to the transaction timestamp, and the value of system-

5. Though SQL:2011 does not define any specific term for such tables, we use the term "bitemporal tables" in keeping with its use in the literature as well as in some products.

time period end column to the highest value of the column's data type.

As in the case of application-time period tables, both the conventional UPDATE statement as well as UPDATE with FOR PORTION OF *app-period*, where *app-period* is the name of application-time period, can be used to modify the rows of bitemporal tables. Similarly, the conventional DELETE statement as well as DELETE with FOR PORTION OF *app-period* can be used to delete rows from bitemporal tables. As in the case of system-versioned tables, only current rows in system-time can be updated or deleted and a historical system row is automatically inserted for every current system row that is updated or deleted.

Queries on bitemporal tables can specify predicates on both application-time periods as well as system-time periods to qualify rows that will be returned as the query result. For example, the following query returns the department where the employee 22217 worked as of December 1, 2010, recorded in the database as of July 1, 2011:

```
SELECT ENo, EDept
FROM Emp FOR SYSTEM_TIME AS OF
   TIMESTAMP '2011-07-01 00:00:00'
WHERE ENo = 22217 AND
 EPeriod CONTAINS DATE '2010-12-01'
```

2.5 Future directions

Though SQL:2011 has incorporated several significant extensions for managing temporal data, there is certainly room for additional extensions. These are left as Language Opportunities for future versions of the standard. Here is a partial list of such extensions:

- Support for period joins, i.e., joining a row from one table with a row from another table such that their application-time or system-time periods satisfy a condition such as overlap. Note that it is possible to do an inner join of this kind using SQL:2011's OVERLAPS predicate, but outer joins require support for additional syntax built into the language.
- Support for period aggregates and period grouped queries that take into account application-time or system-time periods of rows.
- Support for period UNION, INTERSECT and EXCEPT operators that take into account application-time or system-time periods of rows.
- Support for period normalization that produces semantically-equivalent minimal set of rows for a given table by combining contiguous rows that have exactly the same values in non-period columns.
- Support for multiple application-time periods per table.
- Support for non-temporal periods.

3. Comparison with previous temporal proposals

Earlier, we alluded to the fact that the SQL committee had initiated a temporal project that was eventually cancelled around 2001. We list below some of the differences between the approach taken by the previous proposals and the approach taken by SQL:2011 extensions:

- In previous proposals, the period information was associated with the rows of temporal tables using an unnamed hidden column. This design was motivated by the notion of *temporal upward compatibility* [12], which required a temporal table and its equivalent non-temporal table to have exactly the same number of columns. One major drawback of this approach is that it is incompatible with SQL's notion of tables, which requires all information associated with the rows of a table to be captured explicitly as (*and only as*) column values. The other drawback was that queries of the form "SELECT * FROM T", where T is a temporal table, did not return the period information associated with the rows of T in the query result. If users wanted to access the period information associated with the rows, they were forced to include invocations of special built-in functions in the select list of a query for that purpose. These built-in functions operated on the range variables associated with temporal tables in a query expression, and returned the period value associated with the rows pointed to by those range variables. In contrast, the period information is associated with the rows of temporal tables using explicit, user-defined columns in SQL:2011. Also, the period information associated with the rows of a temporal table can be accessed in SQL:2011 simply by including the corresponding period start and end columns in the select list of a query.
- The previous proposals resorted to a controversial technique of prefixing queries, constraints, and insert/update/delete statements with the so-called *statement modifiers* for changing their normal semantics [12]. Unfortunately, previous proposals contained no clear rules specifying the semantics of constructs prefixed with these statement modifiers, so it was hard to figure out the end result [9]. In contrast, SQL:2011 provides a small set of syntactic extensions with clearly-specified scope and semantics.
- In previous proposals, query expressions, constraint definitions, and insert/update/delete statements expressed without the statement modifier prefixes were assumed to operate only on the current rows. This applied to both transaction time tables and valid time tables. While this made sense for transaction time tables, it did not make much sense for valid time tables. For instance, users were allowed to insert into valid time

tables only those rows whose valid time period started with the current time. In fact, there was no way for users to insert rows into valid time tables whose validity periods were either in the past or in the future. In contrast, query expressions, constraint definitions, and insert/update/delete statements on application-time period tables in SQL:2011 operate on the entire table content and follow the standard semantics. Also, SQL:2011 allows users to specify any time values they desire for the application-time period start and end columns as part of the INSERT statement on application-time period tables.

• The previous proposals relied on adding special syntax to the table definition for creating temporal tables (AS TRANSACTION TIME for transaction-time support and AS VALIDTIME for valid-time support). Consequently, supporting additional periods in previous approach would have required extending the table definition syntax every time a new period was added. In contrast, supporting additional periods requires no new syntax in SQL:2011.

4. Acknowledgements

The authors thank Fred Zemke and Matthias Nicola for their valuable comments on the prior versions of this article.

5. References

[1] Fred Zemke, "What's new in SQL:2011", SIGMOD Record, Vol. 41, No. 1, March 2012, pp. 67-73, http://www.sigmod.org/publications/sigmod-record/1203/pdfs/10.industry.zemke.pdf/

[2] Yu Wu, Sushil Jajodia, X. Sean Wang, "Temporal Database Bibliography Update", In Temporal Databases: Research and Practice, O. Etzion, S. Jajodia, and S.Sripada, eds., Springer, 1998

[3] Richard Snodgrass, "Developing Time-Oriented Database Applications in SQL", Morgan Kaufmann, 1999

[4] C. J. Date, Hugh Darwen, Nikos A. Lorentzos, "Temporal Data and the Relational Model", Morgan Kaufman, 2003

[5] Cynthia Saracco, Matthias Nicola, Lenisha Gandhi, "A matter of time: Temporal data management in DB2 10", April 2012, http://www.ibm.com/developerworks/data/library/techarticle/dm-1204db2temporaldata/

[6] Kevin Jerrigan, "Oracle Total Recall with Oracle Database 11g Release 2", September 2009, http://www.oracle.com/us/products/database/security/total-recall-whitepaper-171749.pdf

[7] Gregory Sannik, Fred Daniels, "Enabling the Temporal Data Warehouse", September 2010, http://www.teradata.com/white-papers/

[8] Richard Snodgrass (Ed.), "The TSQL2 Temporal Query Language", Kluwer Academic Publishers, 1995

[9] Hugh Darwen, C.J. Date, "An overview and Analysis of Proposals Based on the TSQL2 Approach", In Date on Database: Writings 2000-2006, C.J. Date, Apress, 2006, also avaliable in http://www.dcs.warwick.ac.uk/~hugh/TTM/OnTSQL2.pdf

[10] ISO/IEC 9075-2:2011, *Information technology—Database languages—SQL—Part 2: Foundation (SQL/Foundation)*, 2011

[11] James F. Allen, "Maintaining knowledge about temporal intervals", Communications of ACM, Vol. 26, No. 11, November 1983

[12] Michael Bohlen, Christian Jensen, Richard Snodgrass, "Temporal Statement Modifiers", ACM Trans. on Database Systems, Vol. 25, No. 4, December 2000

A High-Throughput In-Memory Index, Durable on Flash-based SSD

Insights into the Winning Solution of the SIGMOD Programming Contest 2011

Thomas Kissinger, Benjamin Schlegel, Matthias Boehm*, Dirk Habich, Wolfgang Lehner
Database Technology Group
Dresden University of Technology
01062 Dresden, Germany
{firstname.lastname}@tu-dresden.de

ABSTRACT

Growing memory capacities and the increasing number of cores on modern hardware enforces the design of new in-memory indexing structures that reduce the number of memory transfers and minimizes the need for locking to allow massive parallel access. However, most applications depend on hard durability constraints requiring a persistent medium like SSDs, which shorten the latency and throughput gap between main memory and hard disks. In this paper, we present our winning solution of the SIGMOD Programming Contest 2011. It consists of an in-memory indexing structure that provides a balanced read/write performance as well as non-blocking reads and single-lock writes. Complementary to this index, we describe an SSD-optimized logging approach to fit hard durability requirements at a high throughput rate.

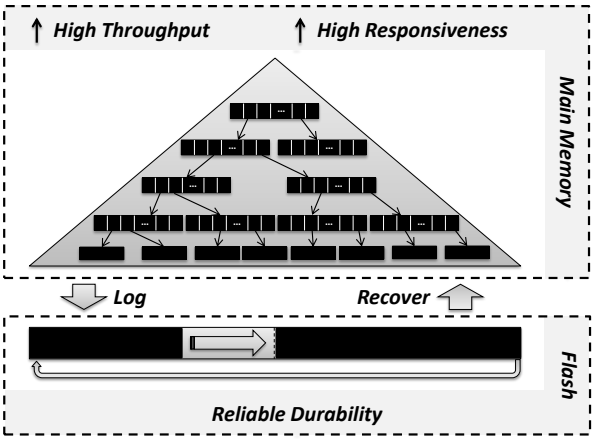


Figure 1: Indexing System Overview.

1. INTRODUCTION

With large main memory capacities becoming affordable over the past years, we observe a shift in the memory hierarchy that degrades hard disks to a persistency-only medium and moves the entire data pool and processing into the main memory. As a second hardware trend, CPU clock rates stopped growing and the number of cores and hardware threads per CPU started to increase constantly. Another present topic are flash-based SSDs, which allow increased throughput and lower latency compared to classic hard disks. When having a look at the application trends, we identify high update rates as a major issue, e.g., in monitoring applications, operational BI or even in social networks. However, common index structures like B+-Trees [2] are designed to work on block-based storage and are not well suited for frequent updates with massive parallelism, because they require complex locking schemes for split and merge operations. This increases communication costs between threads, especially for cache coherency and locking. Furthermore, they are optimized for large block sizes that are used on hard disks. There exist enhancements of the B-Tree that reduce either locking overhead (B-Link Trees [8]) or make them cache-aware (CSB+-Trees [10]). However, these improved structures do not overcome the weaknesses of the B+-Tree base structure in terms of comprehensive balancing tasks and main memory accesses, especially for updates.

In this paper, we describe our solution for the SIGMOD Programming Contest 2011 [1], which addresses exactly the described issues: a high-throughput in-memory index structure, which uses

*The author is currently visiting IBM Almaden Research Center, San Jose, CA, USA.

a flash-based SSD for durability purposes. The task required us to build an in-memory index that fits entirely into the available main memory (two times the total database size) and is able to handle 1024 Byte keys and 4096 Byte values without duplicates. Further, the index needs to offer an order-preserving key-value interface comprising the following operations: (1) *read* the value for a given key, (2) *update* respectively insert a value for a key, (3) *delete* a key-value pair, (4) *compare-and-swap* a value and (5) *iterate* over a key range. The given workload demands a balanced read/write performance as well as a fine-grained locking scheme to allow massive parallel manipulations and reads. For durability, the contest was defining an Intel X25-E enterprise class SSD formatted with ext4. The programming contest constraints granted three times the space of the total database size, which required our solution to perform a continuous garbage collection.

The specifications of the programming contest were released at the end of January 2011 and all teams had about two months available for implementing their solutions. In order to compare the different solutions during this time, the organizers provided a leaderboard, to show the teams each others current results. After the submission deadline was passed, each solution was tested with different workloads (unknown before) to determine the winning team that was finally announced during the SIGMOD 2011.

To give an overview of our winning solution, we illustrate the architecture in Figure 1. The first part of the system forms the index structure that resides completely in the main memory to offer high throughput and low latency to the consuming applications. For our solution, we decided to deploy an enhanced generalized prefix tree [3]. The prefix tree is optimized to work as in-memory structure, because it guarantees a maximum number of memory access for finding a key. In consideration of the workload, this structure also offers a well balanced read/write performance, since updates do not involve neither index node nor memory layout reorganizations. Moreover, its deterministic behavior allows an efficient handling of parallel requests, because there are no costly internal reorganizations that depend on the actual data inside the index.

The contributions of this paper are the presentation of:

1. A fine-grained locking scheme and an efficient memory management subsystem for the generalized prefix tree as index structure, which allows non-blocking reads and single-lock writes. We discuss this locking scheme in Section 2 in more detail.
2. A complementary coalesced cyclic log [7, 6, 5] that is used to log each manipulating operation and to recover the in-memory index in case of hardware or software failure. This log that we describe in Section 3 is tuned to cooperate best with our in memory index structure.

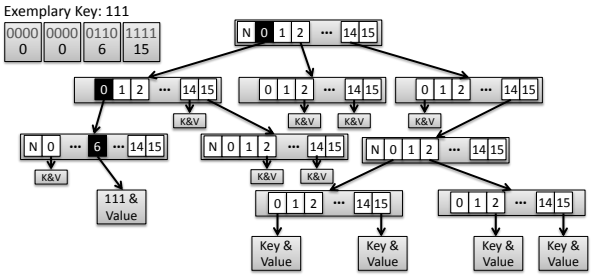


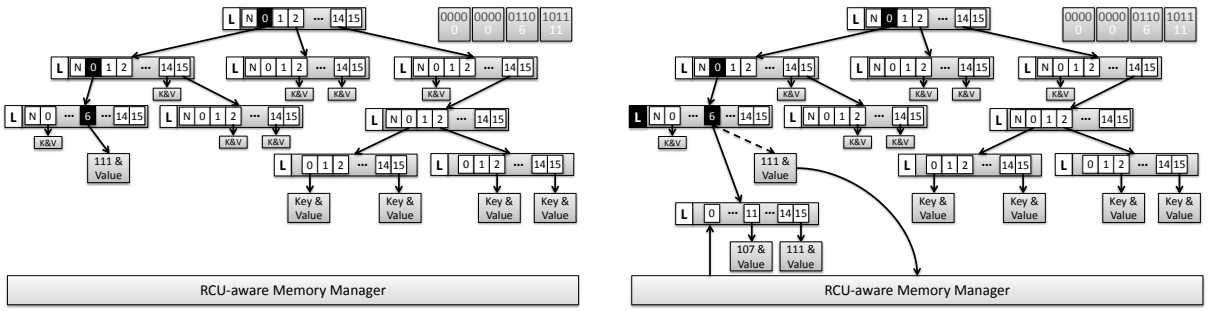
Figure 2: The Generalized Prefix Tree.

After dealing with both system parts, we evaluate each part as well as the overall indexing system in Section 4 on different hardware configurations. Finally, we conclude the paper in Section 5.

2. HIGH-THROUGHPUT IN-MEMORY INDEX

In this section, we start giving an introduction to the generalized prefix tree, followed by the additional changes we made to enable it to handle massive parallel requests. Figure 2 shows an example of a prefix tree in which we highlighted the traversal path for the 16 bit width key 111 (decimal notation). To find a key inside this prefix tree, the key is split into fragments of an equal prefix length k'. Starting from the left, each fragment is used to identify the bucket in the corresponding tree node. For example, the first four bits in this example are used to find the appropriate bucket in the root node for this key. This bucket contains a pointer to the next node that takes the next four bit fragment to find its bucket on this tree level. The number of buckets in each node depends on the prefix length k' and is calculated by $2^{k'}$. At the end of this traversal path is the actual content node, which contains the value for the searched key. So, the main character of a prefix tree is that the key itself is the actual path inside the prefix tree and is independent of other keys present in the index.

The most important configuration parameter is the static prefix length k'. For instance, a 16 bit key ($k = 16$), $k' = 1$ would cause a maximum tree height h of 16 which also leads to 16 costly random



(a) Tree Traversal.

(b) Node Split.

Figure 3: Example of how to insert a new Key into a Prefix Tree.

memory accesses. This configuration is similar to a classical binary tree. The other extreme is $k' = 16$, where only one huge node with 2^{16} buckets is created. Here, we have only one memory access to find a key's value at the cost of a bad memory utilization. For the contest, we set $k' = 4$ to fulfill the memory limitation on the one hand and the performance requirements on the other.

In addition to the base index structure, we applied some performance and memory optimizations as shown in [3]. The *dynamic expansion* expands a node only when a second key with the same prefix is inserted. The example in Figure 2 contains such a case. Key 111's content node is already linked in the third level of the tree. This is possible, because there is no other key inside the tree that uses the same prefix after this point. As soon as a key is inserted that shares the same 12 bit prefix but differs in the fourth fragment, a new node is created at the fourth level. The second optimization we applied is to store the type of the next node directly inside the pointer to that node. So, the highest bit of each pointer (8 Byte aligned memory) determines whether the next node is a content node or an internal node. This reduces the index size and the number of failed speculative execution steps, because the code path is determined much earlier.

The most challenging issue on modern hardware is parallelization. Thus, we are forced to identify a fine-grained locking scheme or even better, to use no locks at all, which is extremely difficult to design and in some cases not even possible. For our solution, we designed a locking scheme that allows non-blocking reads and write operations requiring only a single lock. However, the main bottleneck for the write performance is given through the SSD latency. We will show in Section 3 that due to the special characteristics of the flash-optimized log and the on-the-fly garbage collection, also write operations benefit dramatically from a high degree of parallelism.

In the first place, we describe how to protect write operations against each other. This is achieved by adding a single lock to each internal node. We decided to use spinlocks as the specific locking mechanism, because they (1) have less overhead than mutexes/futexes in their lock and unlock operations and they (2) occupy only 4 Byte of memory, which is much less compared to the size of a mutex structure, which is nearly as big as an entire cache line and therefore doubles the size of each node. Due to the deterministic behavior of our prefix tree, a lookup for a specific key takes always the same path inside the tree and—most importantly—there are no balancing tasks inside and between the internal nodes of the tree. Hence, the nature of the prefix tree allows us to perform a write operation by only locking a single node, because we do not have to lock across multiple nodes for, e.g., balancing purposes. Therefore it is enough to lock the node that needs to be split or where a content node has to be updated. A more fine-grained solution would be to lock single buckets instead of complete nodes, since this would require about 50% more memory for a node, we decided against this solution.

In order to allow non-blocking read operations, we use the read-copy update (RCU) mechanism [9]. With RCU, a content node is never updated in-place by just overwriting the old value with the new one, but it copies the current content node and modifies this new private copy. In a second step, the pointer, which referenced the old content node, is updated to point to the new content node. This allows readers that still read from the old version to finish and takes subsequent readers to the new version of the content node. A problem that arises with RCU is that the memory management needs to detect,

whether it is safe to recycle the old content node's memory block. We accomplished this by adding a counter to each content node that is atomically increased by a reader when starting to read from this content node and is atomically decreased when finished reading the content node. Thus, the RCU-aware memory manager has to test this field for zero, before it can be recycled. The memory manager itself is completely implemented in userland, because `malloc` calls turned out to be much too expensive. Therefore, the memory manager allocates one huge memory chunk via a `mmap` call at the beginning and administrates this chunk on its own. For memory recycling, the memory manager maintains a free list for each possible chunk length, which is limited through the maximum key and value sizes defined by the contest.

EXAMPLE 1. *To summarize, we provide an example in Figure 3. The example shows the write operation of the decimal key 107 (binary representation and fragmentation in the upper right corner of the figure). Compared to Figure 2, every internal node is now extended with a lock. At first, the running thread traverses the prefix tree down to the third level as shown in Figure 3(a). The thread now faces a situation in which the bucket is already occupied by a another content node with another key. Thus, it has to perform a* dynamic expansion *as shown in Figure 3(b). Therefore, it locks the internal node and checks whether the situation is still the same, otherwise it has to retry. At this point it is safe for the thread to work on that internal node. In the next step, the thread asks the memory manager to allocate the new node for the fourth tree level and two new content nodes. The value of the old content node is copied to the new content node and the key tail of the old node is truncated and written to the new one. The content node for the new node is created as usual and both new content nodes are linked by the new internal node. Now, the pointer of the third level node is turned to the new internal node and the node can be unlocked. In a last step, the thread returns the old content block to the memory manager, which is going to recycle its memory as soon as no reader is reading its memory anymore.*

3. COALESCED CYCLIC LOG

In this section, we present our flash-optimized coalesced cyclic log that is tuned to operate hand in hand with the in-memory index structure as depicted in Figure 1. A flash-based SSD basically consists of some flashpacks and a controller. The controller mainly implements the error correction and the wear leveling, which is responsible for prolonging the service life of the flashpacks that are the actual storage media. Main advantages compared to hard disks are better energy-efficiency, energy-proportionality, higher throughput, and lower latency what shortens the gap between transient main-memory and persistent drives. Previous research [4] showed that SSDs expose their full performance when reading or writing to it with a sequential access pattern, similar to hard disks and main memory, because flashpacks are not capable of doing in-place updates. Instead, a SSD has to erase a flash block of typically 4 KB first, before it is able to write this entire block again. Furthermore, flash memory is only able to erase a set of blocks (the erase block size between 128 and 512 KB) at once. All these internal characteristics are hidden from the user through the Flash Translation Layer (FTL). In order to exploit the full performance, we need to be aware of these internal limitations.

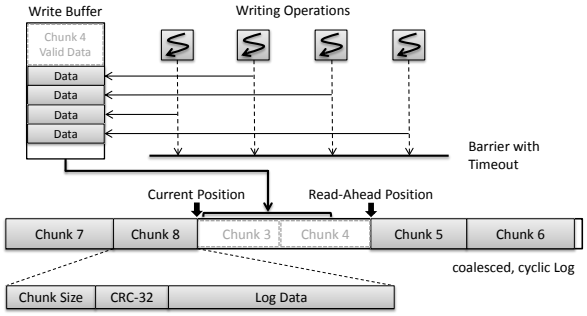


Figure 4: The coalesced cyclic Log.

Since the SSD I/O is the bottleneck of the complete indexing system, it is essential to write with a sequential pattern to the device. Thus, we decided to use an append log as base structure and applied the following two extensions:

1. *Write coalescing* to maximize the write throughput.
2. *Cyclic writing*, because of the limited SSD space.

The idea of *write coalescing* is similar to a group commit. Instead of writing each log record individually, we collect as much as possible log records from the simultaneously running operations in a write buffer and flush them at once. The *write coalescing* increases the overall throughput dramatically, because the contest demanded hard durability, which is ensured by drive cache flushes that are very costly operations with a high latency. As a side effect, this raises the latency of single writing operations. However, it is a good trade-off when taking the throughput gain into account. Figure 4 shows a schematic

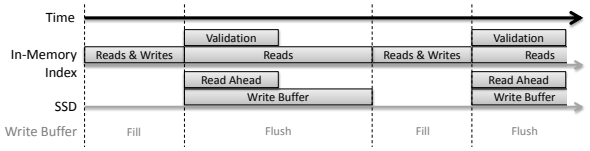


Figure 5: In-Memory Index and SSD Usage over Time.

Operation Type	Probability
Read	45%
Write	40%
Delete	5%
Compare-and-Swap	5%
Scan (max. 10 rows)	5%

Table 1: Distribution of Operation Types.

overview of our coalesced cyclic log. The central component is the described write buffer that collects the single log records from the individual threads. After a thread has written a log record to the write buffer, this thread is stalled until the write buffer is flushed. We flush the write buffer, either it is full, there is no other write operation left, or a predefined timeout is reached. Every time a flush is initiated, the single log records in the write buffer are composed to a chunk that is 4 KB aligned and checksummed with a CRC-32. Afterwards, this chunk is written to the disk using the `fdatasync` system call in Linux.

Since we do not write out the entire index structure as a checkpoint and the available space on the SSD is limited, the log needs to be *cyclic*ally overwritten. This forces us to perform a garbage collection on-the-fly. Thus, the coalesced cyclic log reads at least the size of the write buffer ahead. While reading, it validates the read log records against the in-memory index structure. All log records that are still valid are stored at the beginning of the write buffer and are written together with the new log records on the next write buffer flush.

Figure 5 shows the typical write buffer period and the activity states of the in-memory index as well as of the SSD. At first, when the write buffer is in the *Fill* mode, read and write operations are processed. All changes made by write operations are stored as log data in the write buffer. The corresponding threads are blocked until the write buffer is flushed. As soon as one of the *Flush* conditions for the write buffer occur, the write buffer is locked disabling further write operations. During the write buffer is flushed to the SSD, the storage system already reads ahead the log to free up the space for the next data chunk. The SSD's write performance is not affected by the simultaneous reading, because the operating system usually detects sequential read patterns and prefetches this data in the I/O buffer. After the data was successfully written to the SSD, the write buffer changes back into the *Fill* mode. In order to fully utilize the SSD, it is necessary to keep the time of *Fill* phases as small as possible, because the SSD becomes idle during these times. Thus, we spent a lot of efforts to allow fast parallel operations and non-blocking reads on the in-memory index. Another solution is to use two write buffers and alternately filling and flushing them. However, this solution turned out to be much slower, because of the high latency of a SSD flush operation. Another reason for making reads non-blocking is, that reads are allowed at any time and they are extensively used for log data validation. The main reason for using non-blocking reads was given through the overall scenario: In the contest, the benchmark was setup to create a specific amount of threads. Each thread has a given probability to issue either a read or a write operation and the storage system works optimally when all of these threads flush their writes at once. Therefore, we need to process read operations fast to have every thread doing a write operation to fill the write buffer as fast as possible. This finally prevents the SSD from being idle.

Once, the system crashes or is shutdown, the storage system must be capable of rebuilding the in-memory index. This is done by reading the log twice. The first time, we only process update operations and the second time we apply delete operations. To maintain the temporal order, the in-memory index as well as each log record contains sequential transaction numbers. Thus, an update respectively a delete is only applied, if the transaction number is greater than the current one in the content node of the in-memory index. The need for applying delete operations in a separate run results from this comparison.

4. EVALUATION

In this section, we evaluate individual system components as well as the overall performance on different hardware configurations and parameter settings. The evaluation system, which is different from the system used for the contest, is equipped with an *Intel i7-3960X (6 cores with Hyper-Threading, running at 3.3 GHz and 3.9 GHz max. Turbo Frequency, four memory channels and 15 MB shared L3 cache), 32GB of DDR3-1600, and an Intel X25-E 64GB SSD.* For benchmarking, we

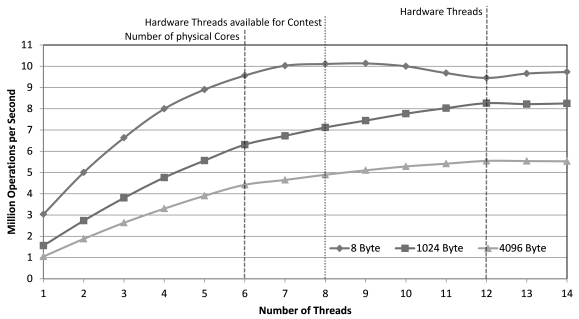


Figure 6: Pure In-Memory Index Performance dependent on the Number of Threads.

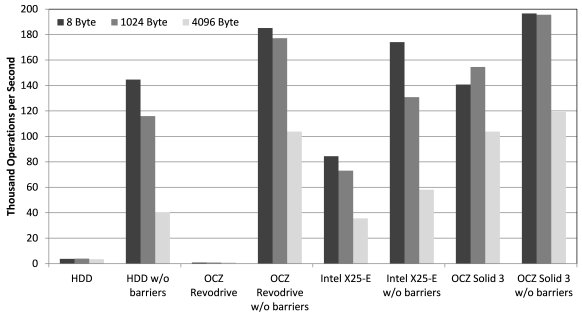


Figure 7: Overall Performance on different Drive Configurations.

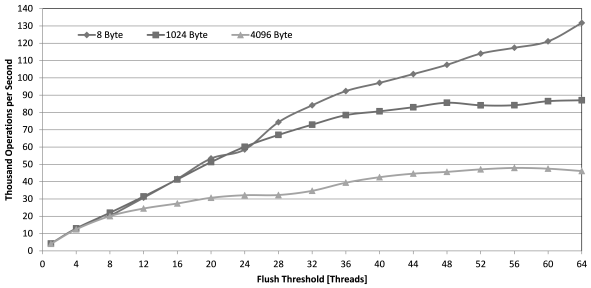


Figure 8: Overall Performance as a Function of the Flush Threshold.

used the benchmark driver provided by the programming contest. This driver generates 8 Byte sequential integer keys to populate the index structure. After that, the driver launches a given set of threads (32 as default) and each of them queries the index for a preset amount of time. Table 1 shows the probabilities for a thread to select a specific type of operation for the next query.

In the first experiment, we evaluate the performance of the pure in-memory index structure. Therefore, we completely disable the SSD log. Figure 6 presents the measurements in million operations per second for a range of 1 million sequential keys (uniformly selected from this range) with a payload of 8, 1024, and 4096 Bytes as values. Further, we marked some points specific to the evaluation hardware. For large payloads like 1024 and 4096 Bytes, we observe optimal scalability of the index. With up to 6 threads, where each of them can be mapped to an exclusive physical core, the performance scales nearly linearly. In the range from 6 to 12 threads, the cores are shared by two threads to fully utilize its processing units, the index still scales nearly linear, but with less gain. The performance gain in this region mainly depends on the remaining amount of memory bandwidth. After the limit of 12 hardware threads is reached, the performance gain stalls and starts to decrease, because of the scheduling overhead. When looking at smaller payloads like 8 Byte, we see another behavior. Here, the performance does not scale linearly, instead, the performance benefit of adding a new thread decreases constantly and even starts to lower the overall performance after nine threads until it reaches the hardware thread limit. This happens because the index is not memory bound anymore and is now facing the high concurrency overhead, especially in the memory management subsystem, when writing a key/value pair to the index. Due to the fact, that the evaluation machine of the contest only got 8 hardware threads available, we did not have any performance penalty by allowing more than eight thread to work in parallel.

In the second experiment, we activated the SSD log and measured the index performance for four different drives. We used an PCI Express OCZ Revodrive, which internally consists of two SSDs connected via a RAID-0 chipset, an Intel X25-E 64GB SATAII enterprise class SSD, and a mainstream OCZ Solid 3 64GB SATA III SSD. Moreover, we tested our solution on a classic Samsung 160 GB SATAII HDD to compare the results with the SSDs. For each drive, we measured the performance for three different payload sizes with and without ext4 barriers. An ext4 barrier guarantees the drive cache to be flushed when calling `fdatasync` (assuming the driver controller supports the `FLUSH CACHE` command). With disabled ext4 barriers, only the file cache is written back to the drives cache. In case of a power loss, the data is not guaranteed to be on the drive. Figure 7 shows all results. In general, we observe major differences between the different drive types with enabled barriers. The worst performance was measured on the Revodrive and the HDD, which are outperformed by orders of magnitude by the X25-E and the Solid 3 drive. With this setting, we mainly measured the latency of the con-

troller respectively the mechanical movements on the HDD, because there is no way of hiding the latency with the drive cache anymore. When turning off the ext4 barriers, we are able to measure the disks bandwidth. Here, the HDD performs much better, because HDD controllers are tuned for latency hiding. There are not much expensive mechanical seeks necessary, because of the sequential write pattern. Furthermore, a HDD is able to overwrite sectors without erasing or copying them first. The best results are achieved with the Revodrive and the Solid 3, which is mainly dedicated to fast PCIe x4 respectively SATAIII interface.

The last experiment, demonstrates the impact of the coalesced writes. This experiment was executed by 64 threads in parallel on an Intel X25-E for different payload sizes and write cache thresholds. For example, a write cache threshold of 4 means that the write cache is immediately flushed after it collected 4 single log records. The respective measurements are visualized in Figure 8. As an overall result, we see that the total performance benefits massively from a high threshold configuration, because the cache flush is the actual bottleneck in the system. When comparing the results for the different payload sizes, we observe that the benefit of flushing more writes at once decreases earlier for big payloads than for the small ones, what can be explained with the SSD hitting its bandwidth limit when transferring the data from the write buffer in the main memory to the drives cache, before it is able to flush this cache.

5. CONCLUSION

With the wide availability of large main memory capacities and multi-core systems, in-memory indexes with efficient parallel access mechanisms become more and more important to databases. Application trends on the other hand, demand hard durability requirements and high update rates, which can not be sustained by classic B-Tree like index structures on conventional hard disks.

Our solution of the SIGMOD Programming Contest 2011, that we presented in this paper, addresses exactly these issues. We designed an indexing structure with an efficient locking scheme that scales with the growing number of hardware threads and exhibits a balanced read/write performance. This structure is based on the generalized prefix tree, which we augmented with an efficient locking scheme to allow non-blocking reads and single-lock writes for fast parallel access. To fulfill durability requirements, we built a storage system that incorporates into this indexing structure and takes advantage of the characteristics of modern flash-based SSDs. The storage system is mainly a cyclic log that collects single log records in a write buffer before flushing it to disk to achieve maximum throughput. The orchestration of both components — the index structure and the storage system — creates a powerful indexing system for modern applications.

6. ACKNOWLEDGMENTS

We thank the NSF, Microsoft, and the ACM for sponsoring this contest and especially the MIT CSAIL for doing such a great job in organizing it. Furthermore, we thank all the other participants for pushing each others solutions forward. This work is supported by the German Research Foundation (DFG) in the Collaborative Research Center 912 “Highly Adaptive Energy-Efficient Computing”.

7. REFERENCES

[1] SIGMOD Programming Contest 2011. `http://db.csail.mit.edu/sigmod11contest/`.

[2] R. Bayer and E. McCreight. *Organization and Maintenance of Large Ordered Indexes*, pages 245–262. Software pioneers, New York, NY, USA, 2002.

[3] M. Böhm, B. Schlegel, P. B. Volk, U. Fischer, D. Habich, and W. Lehner. Efficient In-Memory Indexing with Generalized Prefix Trees. In *BTW*, pages 227–246, 2011.

[4] L. Bouganim, B. T. Jónsson, and P. Bonnet. uFLIP: Understanding Flash IO Patterns. In *CIDR*, 2009.

[5] S. Chen. FlashLogging: Exploiting Flash Devices for Synchronous Logging Performance. In *SIGMOD*, pages 73–86, 2009.

[6] B. K. Debnath, S. Sengupta, and J. Li. FlashStore: High Throughput Persistent Key-Value Store. *PVLDB*, 3(2):1414–1425, 2010.

[7] B. K. Debnath, S. Sengupta, and J. Li. SkimpyStash: RAM Space Skimpy Key-Value Store on Flash-based Storage. In *SIGMOD*, pages 25–36, 2011.

[8] P. L. Lehman and s. B. Yao. Efficient Locking for Concurrent Operations on B-Trees. *ACM Trans. Database Syst.*, 6:650–670, December 1981.

[9] P. E. McKenney and J. D. Slingwine. Read-Copy Update: Using Execution History to Solve Concurrency Problems.

[10] J. Rao and K. A. Ross. Making B+-Trees Cache Conscious in Main Memory. *SIGMOD Rec.*, 29:475–486, May 2000.

Report of the International Workshop on Business Intelligence and the Web – BEWEB 2011

Jose-Norberto Mazón, Irene Garrigós
University of Alicante, Spain
{jnmazon,igarrigos}@dlsi.ua.es

Florian Daniel
University of Trento, Italy
daniel@disi.unitn.it

Malu Castellanos
HP Labs, USA
malu.castellanos@hp.com

ABSTRACT

The 2nd International Workshop on Business intelligencE and the WEB (BEWEB) was co-located with the EDBT/ICDT 2011 Joint Conference in Uppsala (Sweden) on March 25, 2011. BEWEB intends to be an international forum for researchers and practitioners to exchange ideas on how to leverage the huge amount of data that is available on the Web in BI applications and on how to apply Web engineering methods and techniques to the design of BI applications. This report summarizes the 2011 edition of BEWEB.

1. INTRODUCTION

Over the last decade, we have been witnessing an increasing use of Business Intelligence (BI) solutions that allow enterprise to query, understand, and analyze business data in order to make better decisions.

Traditionally, BI applications allowed business people to acquire useful knowledge from the data of their organization by means of a variety of technologies, such as data warehousing, data mining, business performance management, OLAP, periodical business reports, and the like. Yet, in the very recent years, a new trend emerged: BI applications no longer limit their analysis to the data inside a company. Increasingly, they also source their data from the outside, i.e., from the Web, and complement company-internal data with value-adding information from the Web (e.g., retail prices of products sold by competitors), in order to provide richer insights into the dynamics of today's business.

In parallel to the move of data from the Web into BI applications, BI applications are experiencing a trend from company-internal information systems to the cloud: BI as a service (e.g., hosted BI platforms for small- and medium-size companies) is the target of huge investments and the focus of large research efforts by industry and academia. The idea behind BI as a service is outsourcing the processing and analysis of large bodies of data by consuming BI from the cloud, which will help in enabling the so-called Cloud Intelligence [1].

The International Workshop on Business intelligencE and the WEB (BEWEB) targets the above two moves and creates an international forum for exchanging ideas on how to leverage the huge amount of data that is available on the Web in BI applications, and how to apply Web-related engineering methods and techniques to the design of BI applications, such as BI as a service.

BEWEB 2011 attracted an average attendance of about 20 people, who actively engaged in fruitful and animated discussions. High-quality submissions were received from 7 different countries: Austria, Cuba, Germany, Republic of Korea, Mexico, Spain and USA. Each paper was carefully reviewed by at least three members of the program committee. As result of this process, 5 papers were selected as long papers and 1 as short paper for presentation at the workshop. The program included two invited industrial talks and the opening keynote, which was the highlight of the workshop. The proceedings of this year's edition of BEWEB can be found at http://doi.acm.org/10.1145/1966883.

2. KEYNOTE

The keynote entitled "Dr. Crowdsource: or How I Learned to Stop Worrying and Love Web Data" given by Prof. Felix Naumann (Hasso Plattner Institute, Potsdam, Germany) was magnificent. The title was inspired by a famous Stanley Kubrick movie[1], satirizing the threat of nuclear war. Using this metaphor, Prof. Naumann revealed that dealing with web data can be a threat if its inherent heterogeneity is not tackled with appropriate techniques. He described his experiences with the problems caused by web data heterogeneity and the required daunting tasks and state of the art techniques to overcome them, thus facilitating web data integration: source selection to identify appropriate and high-quality sources, data extraction to obtain relevant structured data, scrubbing to standardize and clean data, entity matching to associate different occurrences of the same entity, and, finally, data transformation and data fusion to combine all data about an entity in a single, consistent representation.

3. INVITED TALKS FROM INDUSTRY

With the aim of bringing together researchers from academy and industry, we were proud of having two ani-

[1] http://en.wikipedia.org/wiki/Dr._Strangelove

mated speakers from industrial research labs: Sihem Amer-Yahia (Yahoo! Research) and Xin Luna Dong (AT&T Labs-Research).

Sihem's talk was entitled "I am complex: Cluster Me, Don't Just Rank me". In her lively talk, Sihem argued that Web search over high-dimensional and structured data should go beyond the "10-blue links experience", i.e., a ranked list of results to a keyword-based query. She postulated that an alternative to ranking is to cluster results by means of two approaches: persona-driven search and rank-aware clustering.

In her interesting talk entitled "SOLOMON: Seeking the Truth via Copying Detection", Luna presented the SOLOMON system, whose core detects copying between data sources. She delved into the techniques to effectively detect copying relationships between data sources, leverage the results in various aspects of data integration, and provide a user-friendly interface to facilitate identifying sources that best suit their information needs.

4. RESEARCH SESSIONS

Accepted papers were organized into two research sessions: (i) BI with Web Data and (ii) Engineering Web-Enabled BI.

4.1 BI with Web Data

In the last decade, the amount and complexity of data available on the Web has been growing rapidly. As a consequence, designers of BI applications making use of data from the Web have to deal with several issues. Among the most interesting challenges we find, for instance, the extraction and integration of heterogeneous data sources. But there are many other interesting research challenges that arise when the Web is seen as data repository: developing Web warehousing solutions, tackling data quality issues, leveraging semantic Web technologies, employing Web mining, extending BI to unstructured data (e.g., text) or semi-structured data (e.g., XML), and so on. In addition, Web Intelligence, which explores the use of Artificial Intelligence in conjunction with or in relation to Web technologies, has emerged as a new area that imposes new research challenges. The following three papers were presented in this session:

Self-supervised Web search for any-k complete tuples, by Alexander Löser, Christoph Nagel, Stephan Pieper, Christoph Boden (University of Technology Berlin, Germany). This paper highlights the importance of querying structured information from Web pages. The authors define a query processor that (i) transforms a structured query into a set of keyword queries that are submitted to a search engine; (ii) forwards search results to relation extractors; and then (iii) combines relations into result tuples. This novel query processor completes tuples returned by the relation extractors by systematically discovering any-k relations from Web search results.

Toward total business intelligence incorporating structured and unstructured data, by Byung-Kwon Park (Dong-A University, South Korea), Il-Yeol Song (Drexel University, USA). This paper surveys existing approaches that consolidate both unstructured and structured data for realizing the so-called total business intelligence. After reviewing existing work, the authors present an architecture for total business intelligence in which information retrieval, text mining, and information extraction technologies are integrated with relational OLAP technologies.

Integrating Web feed opinions into a corporate data warehouse, by Lisette García-Moya, Shahad Kudama, Maria Jose Aramburu Cabo, Rafael Berlanga (Universitat Jaume I, Spain). This paper presents an approach to integrate sentiment data extracted from Web opinion feeds into the corporate data warehouse where company analytical data and models are stored. This approach allows BI applications to perform new analysis tasks by using the traditional OLAP-based data warehouse operators.

4.2 Engineering Web-Enabled BI

The move of BI applications from company-internal information systems to applications that are accessible over the Web implies the need for web-specific design competencies. In this context, Web engineering methodologies and technologies represent a large body of knowledge and expertise that could be very useful in the design of applications that allow decision makers to access BI data and functionalities over the Web. Good Web engineering is one of the key foundations in the design of real-time BI and business performance management applications, as Web applications provide access to data from anywhere, at anytime, and via any media. However, BI on the Web implies a plethora of new research challenges that are specific to the BI context, e.g., using Web mashups and RIA for BI development, BI as a service, usability and accessibility for BI applications, etc. This session featured the following three papers:

Capturing data quality requirements for Web applications by means of DQ_WebRE, by César Guerra (UPSLP, Mexico), Ismael Caballero, Mario Piattini (University of Castilla-La Mancha, Spain). This paper presents a model-driven Web engineering approach for considering data quality requirements in the development of Web applications for BI. This approach is based on two artifacts: a metamodel and a UML profile for the management of data quality software requirements for Web applications called DQ_WebRE.

Model-driven restricted-domain adaptation of question answering systems for business intelligence, by Katia Vila (University of Matanzas, Cuba), Antonio Ferrández (University of Alicante, Spain). This paper presents an approach for adapting question answering (QA) systems to restricted domains, such as those related to specific business areas (e.g., healthcare, agriculture, transportation, etc.), with the

aim of providing BI applications with actionable information from unstructured sources (e.g., data from the Web, etc.). QA systems have been applied in an interesting fashion for obtaining concise answers to questions formulated in natural language from a collection of text documents, thus supporting the decision maker in the analysis of textual data sources.

Towards TomTom like systems for the Web: a novel architecture for browser-based mashups, by Emilian Pascalau (Hasso Plattner Institute, University of Potsdam, Germany). This paper introduces a new architecture for browser-based mashups based on the TomTom navigation systems. The described architecture is capable of addressing issues such as BI on demand and instant use, and offers the same degree of generality as the browser.

5. OUTLOOK

In previous editions of BEWEB the focus of the papers has been on synergies between BI and the Web that leverage heterogeneous and semantically rich Web data in BI applications, and use Web-related engineering methods for designing BI as a service. In future editions, we plan to continue this focus but additionally we would like to foster research on methods, models and technologies for realizing the BI-aided Web engineering, i.e., how to acquire, analyze, and manage actionable BI information from Web usage data (e.g., logs, data streams, click streams, etc.) to support the development of Web applications (e.g., to achieve advanced levels of personalization in websites). Likewise, we would like to consider other BEWEB related topics such as big data, data visualization, social networks, streaming data, data quality, privacy and security, and adaptive, contextualized and personalized Web applications.

Finally, to encourage alignment between academic research and industry, we plan to include a session devoted to present innovative industrial products, services, experiences and case studies.

6. ACKNOWLEDGMENTS

We would like to express our gratitude to the Program Committee members for their invaluable work in reviewing the submitted papers, and to the authors for sharing their high quality work and contributing their papers to the workshop proceedings. We are very grateful to Prof. Felix Naumann for accepting our invitation to give a keynote and discussing highly relevant research challenges regarding Web data, and to our invited industrial speakers Sihem Amer-Yahia and Xin Luna Dong. Finally, we would like to thank the EDBT/ICDT Workshops Chair Kjell Orsborn and the Publicity Chair Silvia Stefanova for their support.

7. REFERENCES

[1] Pedersen, T. B. Research challenges for cloud intelligence. In *Proc. of the 1st International Workshop on Business Intelligence and the Web, BEWEB 2010, DOI* = http://doi.acm.org/10.1145/1754239.1754247

SIGMOD 2013 CALL FOR CONTRIBUTIONS

ACM SIGMOD International Conference on Management of Data

New York City, New York, USA
Millennium Broadway Hotel, Times Square
June 23-28, 2013
http://www.sigmod.org/2013/

General Chairs:
Kenneth A. Ross (Columbia University)
Divesh Srivastava (AT&T Labs-Research)
Program Chair:
Dimitris Papadias (HKUST)
Program Committee Group Leaders:
Walid Aref (Purdue)
Amr El Abbadi (UCSB)
Christos Faloutsos (CMU)
Phillip B. Gibbons (Intel Research)
Jayant Haritsa (Indian Institute of Science)
Ihab Ilyas (Qatar Computing Research Institute)
Sam Madden (MIT)
Tamer Oszu (University of Waterloo)
Thomas Seidl (Aachen University)
Vasilis Vassalos (AUEB)
K.Y Whang (KAIST)
Marianne Winslett (UIUC and ADSC)
Jun Yang (Duke)
Keynote and Panel Chair:
Mike Stonebraker (MIT)
Industrial Program Chair:
Nick Koudas (University of Toronto)
Demonstration Chair:
Cyrus Shahabi (USC)
Tutorial Chair:
Yufei Tao (CUHK)
Proceedings Chair:
Stavros Papadopoulos (HKUST)
Workshop Chair:
Christian S. Jensen (Aarhus University)
Undergraduate Research Program Chair:
Alexandra Meliou (UMASS Amherst)
Xiaokui Xiao (NTU Singapore)
Finance Chair:
Graham Cormode (AT&T Labs-Research)
Finance Vice-Chair:
Flip Korn (AT&T Labs-Research)
Publicity/Social Media Chair:
Amelie Marian (Rutgers University)
Sponsorship Chairs:
Divyakant Agrawal (University of California at Santa Barbara)
Dennis Shasha (New York University)
Exhibits Chair:
Mustafa Canim (IBM)
Local Arrangements Chairs:
Cong Yu (Google)
Wendy Hui Wang (Stevens Institute of Technology)
Registration Chair:
Jerome Simeon (IBM)
Demonstration and Workshop Local Arrangements Chairs:
Bishwaranjan Bhattacharjee (IBM)
Tasos Kementsietsidis (IBM)
Web/Information Chair:
Hila Becker (Google)

The annual ACM SIGMOD conference is a leading international forum for database researchers, practitioners, developers, and users to explore cutting-edge ideas and results, and to exchange techniques, tools, and experiences.

SIGMOD 2013 solicits submissions for the programs and events listed below:

- Research papers
- Tutorials
- Panels
- Industrial papers
- Technical demonstrations
- Undergraduate research papers

We invite the submission of original contributions relating to all aspects of data management defined broadly, and particularly encourage submissions on topics of emerging interest in the research and development communities.

Proposals for accompanying workshops are also solicited on topics that demand dedicated coverage due to their relevance to the current data management research and development. The program will also include industrial exhibits, a “New Researcher” symposium, and keynote talks by leaders in academia and industry.

All aspects of the submission and notification process will be handled electronically. Calls for papers and detailed submission information will be available at http://www.sigmod.org/2013/.

New York City is the most populous city in the United States and exerts a significant impact upon global commerce, finance, media, art, fashion, research, technology, education, and entertainment.

The conference will be held at the Millennium Broadway hotel in Times Square. This Art Deco hotel is located in New York City’s renowned Theater District, and is minutes from a host of activities and promises an exciting venue for this year's conference.

SIGMOD 2013 CALL FOR RESEARCH PAPERS

ACM SIGMOD International Conference on Management of Data

New York City, New York, USA
Millennium Broadway Hotel, Times Square
June 23-28, 2013
http://www.sigmod.org/2013/

General Chairs:
Kenneth A. Ross (Columbia University)
Divesh Srivastava (AT&T Labs-Research)
Program Chair:
Dimitris Papadias (HKUST)
Program Committee Group Leaders:
Walid Aref (Purdue)
Amr El Abbadi (UCSB)
Christos Faloutsos (CMU)
Phillip B. Gibbons (Intel Research)
Jayant Haritsa (Indian Institute of Science)
Ihab Ilyas (Qatar Computing Research Institute)
Sam Madden (MIT)
Tamer Oszu (University of Waterloo)
Thomas Seidl (Aachen University)
Vasilis Vassalos (AUEB)
K.Y Whang (KAIST)
Marianne Winslett (UIUC and ADSC)
Jun Yang (Duke)
Keynote and Panel Chair:
Mike Stonebraker (MIT)
Industrial Program Chair:
Nick Koudas (University of Toronto)
Demonstration Chair:
Cyrus Shahabi (USC)
Tutorial Chair:
Yufei Tao (CUHK)
Proceedings Chair:
Stavros Papadopoulos (HKUST)
Workshop Chair:
Christian S. Jensen (Aarhus University)
Undergraduate Research Program Chair:
Alexandra Meliou (UMASS Amherst)
Xiaokui Xiao (NTU Singapore)
Finance Chair:
Graham Cormode (AT&T Labs-Research)
Finance Vice-Chair:
Flip Korn (AT&T Labs-Research)
Publicity/Social Media Chair:
Amelie Marian (Rutgers University)
Sponsorship Chairs:
Divyakant Agrawal (University of California at Santa Barbara)
Dennis Shasha (New York University)
Exhibits Chair:
Mustafa Canim (IBM)
Local Arrangements Chairs:
Cong Yu (Google)
Wendy Hui Wang (Stevens Institute of Technology)
Registration Chair:
Jerome Simeon (IBM)
Demonstration and Workshop Local Arrangements Chairs:
Bishwaranjan Bhattacharjee (IBM)
Tasos Kementsietsidis (IBM)
Web/Information Chair:
Hila Becker (Google)

The annual ACM SIGMOD conference is a leading international forum for database researchers, practitioners, developers, and users to explore cutting-edge ideas and results, and to exchange techniques, tools, and experiences. We invite the submission of original research contributions relating to all aspects of data management defined broadly, and particularly encourage submissions on topics of emerging interest in the research and development communities.

TOPICS OF INTEREST

Topics of interest include but are not limited to the following:

- Storage, Indexing and Physical Database Design
- Query Processing and Optimization
- Text Databases, XML, Keyword Search
- Cloud Computing, Map Reduce, Parallel, Distributed, P2P Systems
- Security, Privacy, Authenticated Query Processing
- Aggregation, Data Warehouses, OLAP, Analytics
- Streams, Sensor Networks, Complex Event Processing
- Knowledge Discovery, Clustering, Data Mining
- Spatial, Temporal, Multimedia and Scientific Databases
- Graph Management, Social Networks
- Systems, Performance, Transaction Processing
- Database Models, Uncertainty, Schema Matching, Data Integration

SUBMISSION GUIDELINES

All aspects of the submission and notification process will be handled electronically. Submissions must adhere to the paper formatting instructions. Research papers will be judged for quality and relevance through double-blind reviewing, where the identities of the authors are withheld from the reviewers. Thus, author names and affiliations must not appear in the papers, and bibliographic references must be adjusted to preserve author anonymity. Submissions should be uploaded at https://cmt.research.microsoft.com/SIGMOD2013/.

For SIGMOD 2013, there will be a "revise and re-submit" option to replace the feedback and shepherding mechanisms of previous conferences. The submissions to be revised will be accompanied with concrete suggestions for improvement, and will go through a second round of reviews.

IMPORTANT DATES

Nov 13, 2012: Research papers due, 5 PM (Pacific Time)
Feb 5, 2013: Notification of acceptance, rejection, revision
Mar 5, 2013: Revised papers due
Apr 9, 2013: Notification of acceptance, rejection for revised papers
Apr 16, 2013: Camera-ready deadline

CALL FOR PAPERS

32nd ACM SIGMOD–SIGACT–SIGART Symposium on

PRINCIPLES OF DATABASE SYSTEMS (PODS 2013)

June 24 - June 26, 2013, New York, New York

Program Chair:

Wenfei Fan
University of Edinburgh
wenfei@inf.ed.ac.uk

Program Committee:

Marcelo Arenas *(PUC)*
Leo Bertossi *(Carleton Univ.)*
Diego Calvanese *(Free Univ. Bolzano)*
Alin Deutsch *(UC San Diego)*
Daniel Deutch *(Ben Gurion Univ.)*
Floris Geerts *(Univ. Antwerp)*
Maurizio Lenzerini *(Rome La Sapienza)*
Benny Kimelfeld *(IBM Almaden)*
Wim Martens *(Univ. Bayreuth)*
Andrew McGregor *(Univ. Massachusetts)*
Frank McSherry *(Microsoft Research)*
Frank Neven *(Hasselt Univ.)*
Jorge Pérez *(Univ. Chile)*
Reinhard Pichler *(Technische Univ. Wien)*
Francesco Scarcello *(Univ. of Calabria)*
Nicole Schweikardt *(Frankfurt Univ.)*
Thomas Schwentick *(TU Dortmund Univ.)*
Peter Widmayer *(ETH)*
Ryan Williams *(Stanford Univ.)*
David Woodruff *(IBM Almaden)*

PODS General Chair:

Richard (Rick) Hull
IBM T.J. Watson Research

Proceedings & Publicity Chair:

Floris Geerts
University of Antwerp

Important Dates:

Short abstracts due:	28 November 2012
Paper submission:	5 December 2012
Notification:	25 February 2013

The PODS symposium series, held in conjunction with the SIGMOD conference series, provides a premier annual forum for the communication of new advances in the theoretical foundations of data management, traditional or non-traditional (see http://www.sigmod.org/the-pods-pages). Topics that fit the interests of the symposium include the following:

big data, alternative query languages, data support for analytics;
query languages for semi-structured data (including XML and RDF);
search query languages (including techniques from information retrieval);
distributed and parallel aspects of data management;
dynamic aspects of databases (updates, views, approximate query answering);
incompleteness, inconsistency, and uncertainty in databases;
schema and query extraction; data integration; data exchange;
provenance; workflows, data-centric Business Process Management;
metadata management; meta-querying; privacy and security;
constraints (specification, reasoning, mining, constraint databases);
Web services; automatic verification of database-driven systems;
model theory, logics, algebras and computational complexity;
data modeling; data structures and algorithms for data management;
design, semantics, and optimization of query and database languages;
domain-specific databases (multi-media, scientific, spatial, temporal, text).

In addition, we *especially welcome papers addressing emerging approaches and challenges in data management*. An External Review Committee will assist in reviewing papers in the following multi-disciplinary areas of *particular interest to this edition of PODS*:

Cloud Computing and Next-generation Distributed Query Processing: Pierre Fraigniaud *(Paris 7)*, Jignesh M. Patel *(Wisconsin)*, Sergei Vassilvitskii *(Google)*, Milan Vojnovic *(Microsoft Research)*

Privacy: Michael Hay *(Cornell)*, Nina Mishra *(Microsoft Research)*, Kobbi Nissim *(Ben-Gurion U.)*, Aaron Roth *(UPenn)*, Adam Smith *(Penn State)*, Mukund Sundararajan *(Google)*

Mining and Learning of Data Models and Queries: Pauli Miettinen *(Max Planck Inst.)*, Evimaria Terzi *(Boston U.)*, Panayiotis Tsaparas *(U. Ioannina)*

Recommendation Systems and Social Networks: Gautam Das *(U. Texas at Arlington)*, Joseph A. Konstan *(U. Minnesota)*

Semantic, Linked, Networked, and Crowdsourced Data: Pascal Hitzler *(Wright State U.)*, David R. Karger *(MIT)*, Boris Motik *(Oxford)*, Neoklis (Alkis) Polyzotis *(UC Santa Cruz)*, Axel Polleres *(Siemens)*

Submitted papers should be at most twelve pages, including bibliography, using reasonable page layout and font size of at least 10pt (note that the SIGMOD style file does not have to be followed). Additional details may be included in an appendix, which, however, will be read at the discretion of the PC. *Papers longer than twelve pages (excluding the appendix) or in font size smaller than 10pt risk rejection without consideration of their merits.* The submission process will be through the Web at *http://www.easychair.org/conferences/?conf=pods2013*. Note that, unlike the SIGMOD conference, PODS does not use double-blind reviewing. The results must be unpublished and not submitted elsewhere, including the formal proceedings of other symposia or workshops. Authors of an accepted paper will be expected to sign copyright release forms, and one author is expected to present it at the conference.

Best Paper Award: An award will be given to the best submission, as judged by the PC.

Best Student Paper Award: There will also be an award for the best submission, as judged by the PC, written by a student or exclusively by students. An author is considered as a student if at the time of submission, the author is enrolled in a program at a university or institution leading to a doctoral/master's/bachelor's degree. The PC reserves the right to give both awards to the same paper, not to give an award, or to split an award among several papers. Papers authored or co-authored by PC members are not eligible for an award.