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:

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

SIGMOD Record Editor:

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

SIGMOD Record Associate Editors:

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

SIGMOD Conference Coordinator:

Sihem Amer-Yahia, Qatar Computing Research Institute, <sihemameryahia AT acm.org>

PODS Executive: Maurizio Lenzerini (Chair), University of Roma 1, <lenzerini AT dis.uniroma1.it>,
Michael Benedikt, Phokion G. Kolaitis, Leonid Libkin, Jan Paradaens and Thomas Schwentick.

Sister Society Liaisons:

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

Awards Committee:

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

Jim Gray Doctoral Dissertation Award Committee:

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

[Last updated : June 15th, 2011]

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)
David DeWitt (1995)
Serge Abiteboul (1998)
Rudolf Bayer (2001)
Ronald Fagin (2004)
Jennifer Widom (2007)
Umeshwar Dayal (2010)
Jim Gray (1993)
C. Mohan (1996)
Hector Garcia-Molina (1999)
Patricia Selinger (2002)
Michael Carey (2005)
Moshe Y. Vardi (2008)
Surajit Chaudhuri (2011)
Philip Bernstein (1994)
David Maier (1997)
Rakesh Agrawal (2000)
Don Chamberlin (2003)
Jeffrey D. Ullman (2006)
Masaru Kitsuregawa (2009)

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)
Jeffrey Ullman (1996)
Raghu Ramakrishnan (1999)
Daniel Rosenkrantz (2001)
Surajit Chaudhuri (2004)
Hans-Jörg Schek (2007)
David Lomet (2010)
Gio Wiederhold (1995)
Avi Silberschatz (1997)
Michael Carey (2000)
Richard Snodgrass (2002)
Hongjun Lu (2005)
Klaus R. Dittrich (2008)
Gerhard Weikum (2011)
Yahiko Kambayashi (1995)
Won Kim (1998)
Laura Haas (2000)
Michael Ley (2003)
Tamer Özsu (2006)
Beng Chin Ooi (2009)

SIGMOD Jim Gray Doctoral Dissertation Award

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

- **2006** *Winner*: Gerome Miklau, University of Washington. *Runners-up*: Marcelo Arenas, 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.

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

[Last updated : June 15th, 2011]

Editor's Notes

From Athens at the end of our annual SIGMOD conference, welcome to the March 2011 issue of the ACM SIGMOD Record!

The first paper of the issue by Ling, Chang and Chao focuses on the efficiency of answering keyword search queries in XML databases. Keyword search is a tempting alternative to structural queries, which are perceived as relatively complex to write and require some knowledge of the XML document organization. Natural candidate answers to an XML keyword search queries are least common ancestors (LCAs) of all the query keywords, however, some of them are intuitively more information-rich and more specific than the others. The algorithms provided in this paper allows speeding up the task of identifying the strongest contributors, that is, the most interesting nodes to return as results of an XML keyword search.

The Systems and Prototypes article by Chen, Kannan, Madhavan and Halevy describes SchemR, a visual search engine and repository for schemas. The motivation for building schema repositories is to simplify the task of designing and deploying a data management application, by enabling would-be user to reuse schemas that others have designed for the same or similar applications. SchemR users can search the repository of schemas by using keywords and/or a partial schema; a comprehensive GUI allows users to navigate the returned results, zoom through the results etc. Importantly, SchemR is part of the OpenII framework, and thus available for other researchers to use.

This issue's Distinguished Database Profile features Dennis Shasha commenting on his many interests, curiosities, stories and insights. As anyone who has read his books (and in particular "Out of their minds") knows, beyond being a great scientist, Dennis is a fascinating storyteller, as this column fully shows. I have had many opportunities to enjoy discussing with Dennis. In INRIA in 2000-2001, when we were office mate, Dennis was presenting us the (then) very new idea of privacy-preserving joins; on a highway crossing primary forests in Brasil, we talked about biology experiment design and how various parts of the world evolved over the centuries. In this column Dennis shares his stories on database tuning, queries by humming, puzzles, and much more.

The Open Forum article by Aumüller and Rahm describes an analysis of recent-years publications in SIGMOD, VLDB, VLDBJ and TODS. Based on DBLP records, the authors analyze the evolution of affiliations of authors by institutions, countries and continents, the structure of international collaborations etc. Following up on a similar SIGMOD Record by M. Nascimento in 2003, this paper provides a quite detailed image of the last 10 years' evolutions in publications in these important venues.

Finally, the issue features three event reports.

The Workshop on Data-Intensive Software Management and Mining was co-located with the ACM CIKM 2010 conference. The main idea behind the gathering lies in four main aspects of the large set of projects whose sources are available today: they can be seen as a web-scale repository, continuously evolving, containing complex-structure items, and forming the basis of a social network of interactions.

The Dagstuhl workshop on Bi-Directional Transformations is a follow-up of a previous conference on the same topic, held in 2008. The workshop reunited researchers from Programming Languages, Graph Transformations, Software Engineering and Database communities in discussions around models for transforming complex (typically graph-structured) data, and their correctness, completeness, possible symmetry, and supported consistency checking.

The Dagstuhl Advanced School on Data Exchange, Data Integration and Streams (DEIS 2010) has been a remarkable innovation over the usual format of a summer school, allowing for much more intensive participation of the attendant PhD students and young researchers. Each of the 22 participants was assigned a topic by the organizers, had to prepare a 45-minutes presentation of the topic and had the opportunity to interact with one organizer while preparing and improving the presentation. This interactive mode was highly appreciated by organizers and participants alike.

I take advantage of these notes to signal you the almost complete renewal of the SIGMOD advisory board, as well as the end of Lisa Singh's tenure as the SIGMOD conference coordinator. Lisa's outstanding service ended, Sihem Amer-Yahia takes over the duty of eliciting, handling and shepherding bids for organizing our yearly conference. The recipients of the 2011 SIGMOD awards have also been added to the list of colleagues we honor in each issue.

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

June 2011

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)

Improving the Performance of Identifying Contributors for XML Keyword Search

Rung-Ren Lin[1], Ya-Hui Chang[2]*, and Kun-Mao Chao[1]
[1]Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan.
{r91054, kmchao}@csie.ntu.edu.tw
[2]Department of Computer Science and Engineering
National Taiwan Ocean University, Keelung, Taiwan.
yahui@ntou.edu.tw

ABSTRACT

Keyword search is a friendly mechanism for users to identify desired information in XML databases, and *LCA* is a popular concept for locating the meaningful subtrees corresponding to query keywords. Among all the LCA-based approaches, MaxMatch [9] is the only one which could achieve the property of *monotonicity* and *consistency*, by outputting only *contributors* instead of the whole subtree. Although the MaxMatch algorithm performs efficiently in some cases, there is still room for improvement. In this paper, we first propose to improve its performance by avoiding unnecessary index accesses. We then speed up the process of *subset detection*, which is a core procedure for determining contributors. The resultant algorithm is called *MinMap* and *MinMap*$^+$, respectively. At last, we analytically and empirically demonstrate the efficiency of our methods. According to our experiments, our two algorithms work better than the existing one, and *MinMap*$^+$ is particularly helpful when the breadth of the tree is large and the number of keywords grows.

1. INTRODUCTION

Keyword search provides a convenient interface for users to obtain desired information from XML documents, but irrelevant data may be returned due to lacking exact query semantics. Therefore, there are a lot of researches on automatically reasoning the meaningful answers for users.

In general, an XML document could be viewed as a rooted tree, where each node represents an element or contents. The LCA-based approaches will identify *the LCA node* first, which contains every keyword under its subtree at least once [2, 4, 5, 6, 7, 12, 13, 14]. Since the LCA nodes sometimes are not very *specific* to users' query, Xu and Papakonstantinou [12] proposed the concept of *SLCA (smallest lowest common ancestor)*, where a node is said to be an SLCA if (i) it is an LCA, and (ii) it has no descendant nodes that also contain all the keywords. For example, consider the XML tree in Figure 1, where each node is associated with a unique Dewey number [15]. For the query Q_0 = (*pitcher, name*), the LCA list is [1, 1.1, 1.1.1, 1.1.3]. Since nodes 1 and 1.1 have descendant nodes 1.1.1 and 1.1.3 that are also LCA, only two nodes, that is, 1.1.1 and 1.1.3, are SLCA. We could see that the two corresponding *player* elements contain more specific information than the elements *players* (1.1) and *team* (1).

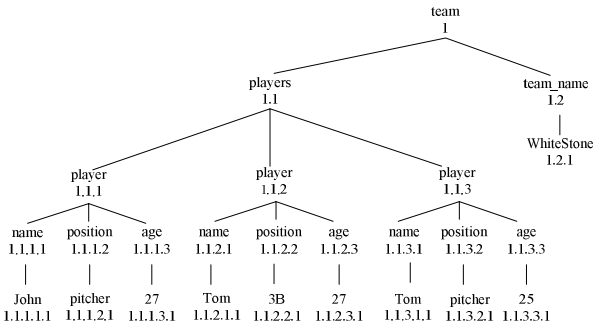


Figure 1: A sample XML tree.

The SLCA approach achieves *specificity* based on the ancestor/decendant relationship, but they do not distinguish the importance of sibling nodes. Therefore, Liu and Chen [9] further proposed the concept *contributor*, where a node is a contributor if it corresponds to more (or equal to) keywords compared with its sibling nodes, and only contributors will be returned. The basic concept of the contributor is to keep only those nodes which contain *richer* information under their subtrees than their siblings. Consider another query Q_1 = (*players, pitcher, Tom*). Since the subtree rooted at node 1.1.2 contains key-

*To whom all correspondence should be sent.

word *Tom* and the subtree rooted at node 1.1.3 contains keywords *Tom* and *pitcher*, node 1.1.3 will be a contributor, and will *prune* node 1.1.2.

One important characteristic of this work is that it satisfies the *monotonicity* and *consistency* properties, which capture a reasonable connection between the new query result and the original query result after an update to the query or to the data. Briefly, the monotonicity property indicates the change to the number of SLCA nodes, and the consistency property describes the change to the content of query result. These properties are sensible and worthwhile, yet none of the other approaches satisfy both properties.

The authors in [9] gave an efficient algorithm *MaxMatch* to locate all the contributors, but there is still room for improvement. First, we identify the places where index accesses are not necessary, and thus avoid unnecessary I/O accesses. Second, we improve the process of *subset detection*, which is a core operation in finding the contributors. We construct the corresponding algorithm MinMap and MinMap$^+$, and perform a series of experiments. Experimental results show that the MaxMatch algorithm is less efficient than our approach when the breadth of the tree is large and the number of keywords grows. Note that the two identified questions are generic and not limited in this framework. Although not major theoretic breakthrough, our findings indeed speedup query processing to a large extent, and can be applied to questions in different domains.

The rest of this paper is organized as follows. In Section 2, we briefly introduce the MaxMatch algorithm [9]. We then present the algorithm MinMap in Section 3, which improves the execution time by avoid unnecessary index accesses. The method for speeding up subset detection is described in Section 4. We further discuss the experimental studies in Section 5. Finally, Section 6 concludes this paper.

2. MAXMATCH

We explain the MaxMatch approach [9] in this section. The sample XML tree given in Figure 1 will be used in the running examples throughout this paper.

We first deliver the definitions given in MaxMatch. A node is a *match* if its tag name or the content corresponds to a given query keyword. The *descendant matches* of a node n, denoted as $dMatch(n)$, are a set of query keywords, each of which has at least one match in the subtree rooted at n. In addition, a node n is a *contributor* if (i) n is the descendant of a given SLCA or n itself is one of the SLCAs, and (ii) n does not have a sibling n' such that $dMatch(n') \supset dMatch(n)$. For each SLCA, the MaxMatch will return the root-to-a-match path, as long as the nodes in the path are all contributors.

Take query Q_1 as an example again. We have $dMatch(1.1.1) = \{pitcher\}$, $dMatch(1.1.2) = \{Tom\}$, and $dMatch(1.1.3) = \{pitcher, Tom\}$. According to the definitions mentioned above, nodes 1.1.1 and 1.1.2 are not contributors since both $dMatch(1.1.1)$ and $dMatch(1.1.2)$ are the proper subsets of $dMatch(1.1.3)$. The output will be the node list: [1.1 (*players*), 1.1.3 (*player*), 1.1.3.1 (*name*), 1.1.3.1.1 (*Tom*), 1.1.3.2 (*position*), 1.1.3.2.1 (*pitcher*)].

Observe that the second condition of the contributor involves *subset detection*. Given a node n_p with b children, the naive algorithm, which compares all possible pairs among those b children, takes $O(b^2)$ time and is time-consuming. To promote the efficiency, MaxMatch uses a boolean array $dMatchSet$ to record the information of each of n_p's child. Specifically, let $S = \{n_1, n_2, ..., n_b\}$ be the children set of n_p. The $dMatchSet$ array for n_p is of length 2^w, where w is the number of keywords. All the bits are initialized as $false$ at the beginning. The j^{th} bit, $0 \leq j \leq 2^w - 1$, is set to *true* if and only if n_p has at least one child $n_i \in S$ such that the decimal value of $dMatch(n_i)$ is j. Then, for each $n_i \in S$, MaxMatch can determine whether it is a contributor or not in $O(2^w)$ time by scanning the whole $dMatchSet$ array. Hence, it totally takes $O(b \cdot 2^w)$ time to deal with all the b children. Note that the number of query keywords w is generally small, and the branch factor b may be very large in the XML trees. Therefore, under the condition of $b \gg w$ (such as $b > 2^w$), MaxMatch works better than the naive algorithm.

Although MaxMatch is quite efficient, we will propose a more efficient way to perform subset detection which is described in Section 4. In addition, while setting the $dMatch$ values, MaxMatch retrieves the tag names of a node using the index. Since some of the nodes are pruned at last, it may cause redundant I/O accesses. We will propose an improved algorithm in Section 3.

3. AVOIDING INDEX ACCESSES

We first introduce the definitions of our approach. The *match tree* of a node t, denoted as $mTree(t)$, consists of the nodes along the path from each match up to t. The nodes in the match tree without matching any query keyword are called *non-keyword nodes*. Besides, node n is called a *hit node* if (i) n is contained in the match tree rooted at a given SLCA, (ii) n is a non-keyword node, and (iii) all the nodes

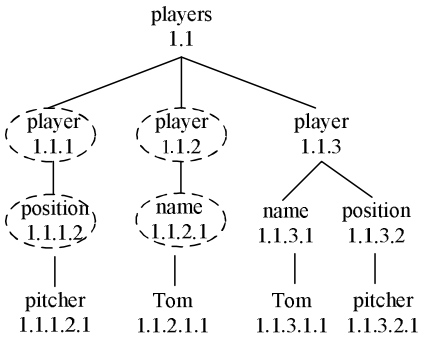


Figure 2: The match tree of Q_1.

on the path from n up to the SLCA are contributors. On the contrary, node n is called a *miss node* if n satisfies the first two conditions of hit node, but does not satisfy the third condition. Consider query Q_1 again. Recall that nodes 1.1.1 and node 1.1.2 are pruned by node 1.1.3. Therefore, the hit node list is [1.1.3, 1.1.3.1, 1.1.3.2], and the miss node list is [1.1.1, 1.1.1.2, 1.1.2, 1.1.2.1], where miss nodes are depicted by dashed circles as shown in Figure 2.

Recall that MaxMatch retrieves the tag names of nodes using indexes, but some of them are eventually pruned. In our approach, we let every node directly inherit the $dMatch$ information from its children without retrieving its own tag name, and use a variable to record if it is a non-keyword node. We retrieve its tag name only when it is confirmed to be a contributor, and thus save unnecessary index accesses. Interested readers can refer to [8] for the complete algorithm. We further define the *miss rate* as: Σ miss nodes /(Σ hit nodes + Σ miss nodes) for match trees of the SLCAs. As an example, the miss rate in Figure 2 is $4/(3+4) \approx 57\%$. Therefore, our approach will save up to 57% index accesses compared with the MaxMatch when constructing non-keyword part of the match tree. Later, the miss rate will be used for analyzing the processing time between MinMap and MaxMatch, and we will have more details in Section 5.

4. IMPROVING SUBSET DETECTION

Recall that MaxMatch has the time complexity $O(b \cdot 2^w)$ when performing subset detection. In this section, we propose an $O(b) + O(w \cdot 2^w)$-time method, to speed up the process when the breadth of the tree is large and the number of keywords grows.

4.1 The Algorithm

Consider a parent node in $mTree(t)$ with b children $\{n_1, n_2, ..., n_b\}$ and their descendant matches

Input: a set of b nodes $N = \{n_1, n_2, \ldots, n_b\}$ and their descendant matches $D = \{dMatch(n_1), dMatch(n_2), \ldots, dMatch(n_b)\}$. Suppose the number of keywords is w.

Output: all the nodes in N that are not pruned by any of their siblings.

FADC(*D*)
1: allocate an array A of length 2^w
2: set every element of A to *empty*
3: **for** each $dMatch(n_i) \in D$ $(1 \le i \le b)$ **do**
4: $k \leftarrow num(dMatch(n_i))$
5: **if** $A[k]$ = *empty* **then**
6: $A[k] \leftarrow$ *equal*
7: *SetSubset*($dMatch(n_i)$)
8: **for** each $dMatch(n_i) \in D$ $(1 \le i \le b)$ **do**
9: $k \leftarrow num(dMatch(n_i))$
10: **if** $A[k]$ = *subset* **then**
11: prune n_i
12: output all of the unpruned nodes

SetSubset(*d*)
1: **for** each max-subset d' of d **do**
2: $k \leftarrow num(d')$
3: **if** $A[k]$ = *empty* **then**
4: *SetSubset*(d')
5: $A[k] \leftarrow$ *subset*

Figure 3: The algorithm for improving subset detection.

$D = \{dMatch(n_1), ..., dMatch(n_b)\}$. The problem of determining contributors is to prune those nodes whose $dMatch$ sets are the proper sets of those of their siblings.

Suppose query Q has w keywords. There are totally 2^w distinct subsets of Q. We propose to classify each distinct subset d of Q into three *states* as follows:

- *empty*: There does not exist any $dMatch(n_i) \in D$ such that $d \subseteq dMatch(n_i)$.
- *subset*: There exists at least one $dMatch(n_i) \in D$ such that $d \subset dMatch(n_i)$.
- *equal*: There does not exist any $dMatch(n_i) \in D$ such that $d \subset dMatch(n_i)$. However, there exists at least one $dMatch(n_i) \in D$ such that $d = dMatch(n_i)$.

The main idea of our approach is to set the states of all the subsets of each $dMatch(n_i) \in D$, to facilitate later processing. To achieve this purpose, an array of length 2^w is used to record the state of every distinct subset of query Q, where w is the number of keywords of Q. We also define the concept of *max-subset*. Set d' is said to be a max-subset of set d if (i) $d' \subset d$, and (ii) $|d'| = |d| - 1$, where $|d|$ and $|d'|$ represent the numbers of keywords in sets $|d|$ and $|d'|$, respectively. It is obvious that d has exactly $|d|$ distinct max-subsets.

Figure 3 shows the complete algorithm. The input is a set of sets D, and every set in D is denoted as $dMatch(n)$ which represents the keywords that are contained in the subtree of node n.

0000	subset	1000	empty
0001	empty	1001	empty
0010	subset	1010	empty
0011	empty	1011	empty
0100	subset	1100	empty
0101	empty	1101	empty
0110	equal	1110	empty
0111	empty	1111	empty

(a) After $dMatch$(1.1.1) is processed

0000	subset	1000	subset
0001	empty	1001	empty
0010	subset	1010	subset
0011	empty	1011	empty
0100	subset	1100	subset
0101	empty	1101	empty
0110	subset	1110	equal
0111	empty	1111	empty

(b) After $dMatch$(1.1.3) is processed

Figure 4: The state-array of query Q_2.

At first, we allocate an array A of length 2^w to record the state of each distinct subset of Q. Each element of A is set to *empty* at the beginning in line 2. The *num* function (in line 4) transfers the $dMatch(n_i)$ into a decimal value. It keeps a boolean array of length w to record the keywords contained in $dMatch(n_i)$ set and then transfers the boolean array into a decimal value. Specifically, the j^{th} bit of this boolean array is set to *true* if $dMatch(n_i)$ contains the j^{th} keyword of Q. Let $k = num(dMatch(n_i))$. We check the state of $A[k]$ for each $dMatch(n_i) \in D$. If $A[k]$ is *empty* in line 5, it means that so far none of the sets from $dMatch(n_1)$ to $dMatch(n_{i-1})$ equal to $dMatch(n_i)$ or are the supersets of $dMatch(n_i)$. We then set $A[k]$ to *equal* and call procedure $SetSubset(dMatch(n_i))$ to set the states of $dMatch(n_i)$'s max-subsets. On the contrary, if $A[k]$ is not *empty* in line 5, we skip $dMatch(n_i)$ no matter the state of $A[k]$ is *equal* or *subset*. Because it implies that there may exist one or more sets from $dMatch(n_1)$ to $dMatch(n_{i-1})$ that equal to $dMatch(n_i)$ or are the supersets of $dMatch(n_i)$.

Procedure *SetSubset* takes a set d as the parameter, and recursively calls itself by sending every max-subset d' of d as the parameter if $A[k] =$ *empty*, where k is the decimal value of d' in Procedure *SetSubset*. After processing each $dMatch(n_i) \in D$ (lines 3 to 7), the state of $A[num(dMatch(n_i))]$ $(1 \leq i \leq b)$ is either *equal* or *subset*. Clearly, node n_i should be pruned if the state of $A[num(dMatch(n_i))]$ is eventually set to *subset*. Therefore, we can obtain all of the contributors (unpruned nodes) easily.

Example 1. Consider query Q_2 = (*25, pitcher, name, players*). Node 1.1 is the only SLCA of Q_2, and we are going to determine the contributors among nodes 1.1.1, 1.1.2, and 1.1.3. Suppose the first keyword (from left to right) corresponds to the leftmost bit. Therefore, the $num(dMatch)$ values of these three nodes are 0110_{bin}, 0010_{bin}, and 1110_{bin}, respectively. At first, an array A of length 16 (2^4) is allocated, and each $A[k]$ $(0 \leq k \leq 15)$ is assigned as *empty*. Consider the first input $dMatch$(1.1.1). Since $A[0110_{bin}]$ is *empty*, we set $A[0110_{bin}]$ to *equal* and then call procedure *SetSubset* by parameter $dMatch$(1.1.1). Accordingly, $A[0010_{bin}]$, $A[0100_{bin}]$, and $A[0000_{bin}]$ are recursively set to *subset* by Procedure *SetSubset* (as shown in Figure 4 (a)). The second input is $dMatch$(1.1.2). Clearly, nothing is changed since the state of $A[0010_{bin}]$ is not *empty*.

At last, consider the third input $dMatch$(1.1.3). We first set $A[1110_{bin}]$ to *equal* and then call procedure *SetSubset* to set the max-subsets of $dMatch$(1.1.3). Note that $A[0110_{bin}]$ is set to *equal* by $dMatch$(1.1.1), and now it would be changed to *subset*. We skip discussing the latter two max-subsets of $dMatch$(1.1.3), because they are quite similar to that of the first input $dMatch$(1.1.1). In summary, the final state-array of Q_2 is shown in Figure 4 (b). Since $A[0110_{bin}]$ and $A[0010_{bin}]$ are both set to *subset*, nodes 1.1.1 and 1.1.2 are pruned eventually. □

4.2 Time Complexity

Procedure *SetSubset* is called only when the state of a given $A[k]$ is *empty*, and then $A[k]$ will be set to *subset*. Obviously, procedure *SetSubset* is repeated at most 2^w times. In addition, for each call of procedure *SetSubset*, it takes $O(w)$ time to prepare all the max-subsets. Therefore, procedure *SetSubset* costs $O(w \cdot 2^w)$ during the whole process. At last, it takes $O(b)$ time to determine all of the pruned elements, so the time complexity is $O(b) + O(w \cdot 2^w)$ in total.

5. EXPERIMENTAL STUDIES

We have implemented the approached described in Section 3, and call the system MinMap. We have also included the approach described in Section 4 in MinMap and call the resultant algorithm MinMap$^+$. Therefore, the only difference between MinMap and MinMap$^+$ is in processing subset detection. All these algorithms are implemented in C++ with the environment of Windows XP and Visual Studio 6.0. In addition, the two proposed systems utilized two indices based on B-tree structure, similar to MaxMatch.[1] These indices are created and accessed based on the Oracle Berkeley DB [16].

5.1 Comparing MaxMatch with Our Approaches

In this section, we will compare the processing time of MinMap, MinMap$^+$, and MaxMatch. The

[1]In the first index, the key is the Dewey number of a node and returns the corresponding tag name. In the second index, the data associated with each keyword k is a sorted list of Dewey numbers of all the nodes which contain keyword k.

Table 1: Test queries for DBLP data set.

No.	query	FQ	MR
QD_1	1998, volume, 1	HHH	1%
QD_2	1998, LNCS, volume, 1	HLHH	98%
QD_3	JACM, cite, On the complexity of integer programming	LHL	35%
QD_4	2001, FOCS, article	HLH	84%
QD_5	2000, Springer, editor	HLH	0%
QD_6	1999, cdrom, Information Processing Letters	HHL	97%

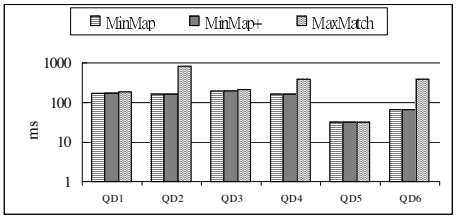
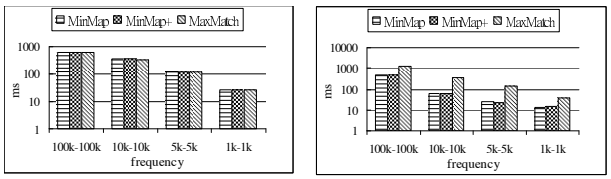


Figure 5: Processing time of the test queries in Table 1.

DBLP data set is used to perform the experiments. We start by designing queries based on different combination of keywords with high-frequency and/or low-frequency. Table 1 shows the testing queries. The keyword frequencies (denoted as FQ) are displayed in the third column, where "H" stands for *high* and "L" stands for *low*. The miss rate of the queries (denoted as MR) are also specified in the last column.

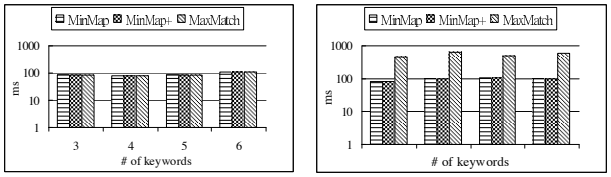
The results of processing time are shown in Figure 5. We can see that keyword frequencies have no direct impact on the performance of the three systems. For example, QD_4, QD_5, and QD_6 all consist of two high-frequency keywords and one low-frequency keyword. However, MaxMatch performs the same as our two proposed systems for QD_5, but performs worse for QD_4 and QD_6. The reason is that the elements matching the querying keywords of QD_5, that is, *year*, *publisher*, and *editor*, all reside under the same parent elements, which makes the miss rate low. In contrast, among the elements matching the querying keywords of QD_4, *article* is the parent of *booktitle* and *year*, which makes the miss rate high. However, we can observe that both MinMap and MinMap$^+$ work better than MaxMatch for all those high miss-rate queries.

In the next experiment, we specifically control the frequencies of keywords to identify their relationship with the miss rate and examine how they affect the performance. We keep the minimum frequency and the maximum frequency of the keywords to be close, and vary the frequencies simultaneously. Therefore, all the keywords in the same query have similar frequencies. As shown in Figure 6, observe again that the keyword frequency do not show direct impact on performance. However, our approaches perform a lot better than MaxMatch when the miss rate is high (Figure 6(b)), and the performance is even to an order of magnitude difference. It also shows that the miss rate has no obvious relationship with keyword frequencies alone.



(a) low-miss rate (DBLP data set) (b) high-miss rate (DBLP data set)

Figure 6: Varying the keyword frequencies.



(a) low-miss rate (DBLP data set) (b) high-miss rate (DBLP data set)

Figure 7: Varying the number of keywords.

Finally, we design different scenarios by changing the number of keywords. We fix the maximum frequency of the keyword and perform random testing. We then classify all the experimental results into low miss-rate (smaller than 10%) cases and high miss-rate (larger than 90%) cases. The experimental results depicted in Figure 7 show the similar result to the previous scenario.

We have also applied the other two data sets to perform the experiments: SwissProt.xml[2] and baseball.xml[3]. The testing results are similar to that of DBLP data set. Due to space limitation, we omit the experimental results.

5.2 Comparing MinMap and MinMap$^+$

We then compare MinMap with MinMap$^+$ in this subsection. Recall that MinMap applies the subset detection method in the original MaxMatch algorithm, while MinMap$^+$ applies the new method proposed in Section 4. Also recall that their time complexity is affected by the branch factor b and the number of keyword w. We use several datasets with different branch factor to perform the test. We then examine how the number of keywords affect

[2]http://www.cs.washington.edu/research/xmldatasets/.
[3]http://www.cafeconleche.org/books/biblegold/examples/.

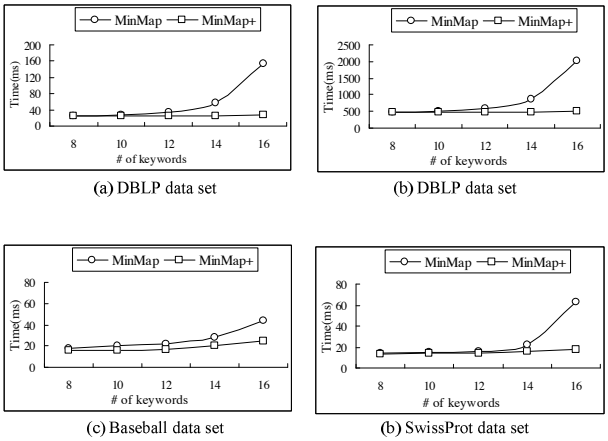


Figure 8: Comparing MinMap and MinMap$^+$.

the performance. We first apply the DBLP dataset. In Figure 8 (a)-(b), the maximum branch factors of the match trees are about 5,000 and 50,000, respectively. We also make the total frequencies of the keywords not change too much when the number of keywords grows. The result shows that the processing time of MinMap increases sharply while the processing time of MinMap$^+$ increases smoothly along with the number of keyword in both cases.

We then apply the Baseball and SwissProt data sets to perform the similar experiment. In Figure 8 (c)-(d), the maximum branch factors are about 50, and 7,000, respectively. Since the maximum branch factor of Figure 8 (c) is small compared with the other three, the improvement between MinMap and MinMap$^+$ is therefore not that large. However, MinMap$^+$ still outperforms MinMap.

6. CONCLUSIONS

In this paper, we propose two algorithms to improve the efficiency of MaxMatch. The MinMap algorithm is designed based on eliminating unnecessary index accesses during the construction of the match tree. The MinMap$^+$ algorithm is proposed to speed up the computation of subset detection. The experimental results show that MinMap outperforms MaxMatch when the miss rate is high. The experiments also show that MinMap$^+$ is particularly helpful when the breadth of the tree is large and the number of keywords grows. As part of our future work, we are interested in designing a novel ranking scheme to order the query results so that users may focus on the most desirable ones.

7. REFERENCES

[1] B. Cate and C. Lutz. The Complexity of Query Containment in Expressive Fragments of XPath 2.0. In *Journal of ACM*, pages 73-82, 2009.

[2] S. Cohen, J.Mamou, Y.Kanza, and Y.Sagiv. XSEarch: A Semantic Search Engine for XML. In *VLDB*, pages 45-56, 2003.

[3] S. Flesca, F. Furfaro, and S. Greco. A Query Language for XML Based on Graph Grammars. In *Journal of WWW*, pages 125-157, 2002.

[4] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: Ranked Keyword Search over XML Documents. In *SIGMOD*, pages 16-27, 2003.

[5] G. Koloniari and E. Pitoura. LCA-Based Selection for XML Document Collections. In *WWW*, pages 511-520, 2010.

[6] G. Li, J. Feng, J. Wang, and L. Zhou. Effective Keyword Search for Valuable LCAs over XML Documents. In *CIKM*, pages 31-40, 2007.

[7] Y. Li, C. Yu, and H. V. Jagadish. Schema-Free XQuery. In *VLDB*, pages 72-83, 2004.

[8] R. R. Lin, Y. H. Chang, K. M. Chao. Faster Algorithms for Searching Relevant Matches in XML Databases. In *DEXA* (LNCS 6261), pages 290-297, 2010.

[9] Z. Liu and Y. Chen. Reasoning and Identifying Relevant Matches for XML Keyword Search. In *PVLDB*, pages 921-932, 2008.

[10] I. Tatarinov, S. Viglas, K. Beyer, J. Shanmugasundaram, E. Shekita, and C. Zhang. Storing and Querying Ordered XML Using a Relational Database System. In *SIGMOD*, pages 204-215, 2002.

[11] X. Wu, D. Theodoratos, S. Souldatos, T. Dalamagas, and T. Sellis Evaluation Techniques for Generalized Path Pattern Queries on XML Data. In *Journal of WWW*, pages 441-474, 2010.

[12] Y. Xu and Y. Papakonstantinou. Efficient Keyword Search for Smallest LCAs in XML Databases. In *SIGMOD*, pages 527-538, 2005.

[13] Y. Xu and Y. Papakonstantinou. Efficient lca based keyword search in xml data. In *EDBT*, pages 535-546, 2008.

[14] R. Zhou, C. Liu, and J. Li. Fast ELCA Computation for Keyword Queries on XML Data. In *EDBT*, pages 549-560, 2010.

[15] Dewey Decimal Classification. http://www.oclc.org/dewey/.

[16] Oracle Berkeley DB. http://www.oracle.com/technetwork/database/.

Exploring Schema Repositories with Schemr

Kuang Chen and Akshay Kannan
University of California, Berkeley
kuangc@cs.berkeley.edu,
akannan@cs.berkeley.edu

Jayant Madhavan and Alon Halevy
Google, Inc.
jayant@google.com,
halevy@google.com

ABSTRACT

Schemr is a search engine for users to search for and visualize schemas in a metadata repository. Users may search by keywords and by example, using schema fragments as query terms. Schemr uses a novel search algorithm, based on a combination of text search and schema matching techniques, coupled with a structurally-aware scoring metric. Schemr presents search results in a GUI that allows users to explore which elements match and how well they do. The GUI supports interactions, including panning, zooming, layout and drilling-in. This paper introduces Schemr as a new component of the information integration toolbox and discusses its benefits in several applications.

1. INTRODUCTION

All around the world, groups of small organizations want to share structured data with each other. For instance, consider the Nature Conservancy's[1] efforts in rallying small conservation organizations to contribute environmental monitoring data. As another example, consider a rural health system in sub-Saharan Africa, consisting of community health workers, low-resource health clinics, and district-, regional-, and national-level ministries of health. Unlike typical data integration scenarios, where the goal is to provide uniform access to multiple *existing* data sets, here organizations are more willing to share information right from the beginning. In particular, a database designer working on a new schema is likely to consult and explore existing schemata, given access to them. What is needed is a tool for schema search and visualization to guide the initial database design.

This paper describes Schemr, a tool for locating, exploring, and reusing relevant schemas (or schema fragments) in large schema collections. Schemr leverages the past experience of a collaborative community and the algorithmic techniques from existing information integration tools to lower the data sharing barrier-to-entry and nurture schema compatibility – simplifying the task of information integration during schema design. Beyond the example scenarios described above, a schema search tool is a valuable tool to navigate any dataspace of heterogeneous information [4].

Schemr's search algorithm combines schema matching and text search techniques with a structurally-aware scoring metric. Designers can use keywords or existing schema fragments as search terms. Results are returned in an environment that allows users to visually explore and compare matching schemata.

Specifically, we make the following contributions:

- Schema search algorithm - Schemr's search algorithm combines techniques from document search and schema matching, and employs a holistic *tightness-of-fit* measure to find and rank schemas according to a query's semantic intent.
- Visualizations of results - Schemr visualizes search results in an interactive web application, allowing users to compare multiple results and drill-in to explore a schema with visually encoded similarity measures.
- Open source software component - Schemr is part of the OpenII open-source information integration framework [8], which any organization may use and extend for free.

A demonstration of Schemr was first presented at SIGMOD 2009 [2].

Example Scenario

We ground our motivation through discussion and observation of two such organizations, mentioned above: the Nature Conservancy and a large HIV/AIDS treatment program in Tanzania. We found that data management in these organizations takes place in an ad-hoc manner, with ad-hoc tools. Data administrators

[1]The Nature Conservancy http://www.nature.org

face a vicious cycle: they are overloaded with requests to manually curate data that should be produced by automated processes. Thus, having no time to tackle major system improvements, they create stopgap solutions. These data administrators say that they would gladly collaborate with others to share schemas and advice, but are hindered by the high-maintenance cost of the stopgap solutions. They need tools that provide an immediate productivity gain. For these organizations, sharing designs through schema search can provide this bootstrap path, which starts with better data modeling and leads to better integrated information.

In such a setting, the database administrator begins by designing a new table. She is unsure of the best way to model the table, and wants to search for related schemas and data examples. She opens Schemr in her web browser and has the option to specify either a simple keyword search and/or a partially designed schema fragment. In this case, let us suppose that she performs a search for existing data models by using the keywords `patient, height, gender, diagnosis`. Additionally, she specifies a partially designed schema to specify results that include elements semantically equivalent to ones she has already designed. A partially designed schema can be specified by uploading a *DDL* (Data Definition Language) or *XSD* (XML Schema Definition).

Upon executing this query, the designer is presented with several relevant schemata to explore in further depth. The results can come from a variety of sources: reference schemata within the organization, shared schemata from partnering organizations, or public sources.

Internally, Schemr parses the input schema and creates a *query graph* (Figure 1) out of it, on which the similarity functions are computed. Schemr returns a ranked list of n results, presented in a tabular format, including columns for name, score, matches, entities, attributes, and description. The user can interact with the results by clicking on a particular entry to visualize its schema elements, or ask for the next n schemas. On drill-in to a particular schema, Schemr creates a detailed graph structure with visual encodings of similarity. Figure 2 shows an example of Schemr's visualizations.

2. SCHEMR OVERVIEW

In this section, we first describe Schemr's search algorithm, and then describe our implementation.

Algorithm

Schemr's search algorithm (Figure 3) consists of three phases. Prior to executing a search, the query parser creates a query-graph from the keyword terms and schema fragments given by user input. In the first phase, *Candidate Extraction*, Schemr flattens the query-graph into a list of keywords and quickly retrieves the top candidate schemas from a scalable document index. In the second phase, *Schema Matching*, Schemr evaluates the top candidate schemas with an ensemble of schema matchers [3, 6], scoring the semantic similarity between candidate schema and the query-graph elements. In the third phase, Schemr computes a final score by weighing similarity scores with a *Tightness-of-fit Measurement*.

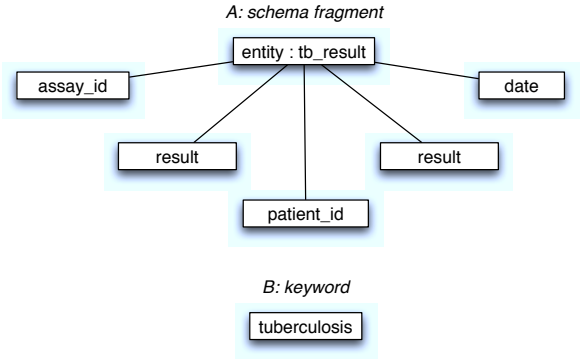


Figure 1: An example query graph consisting of both (A) a schema fragment and (B) a keyword

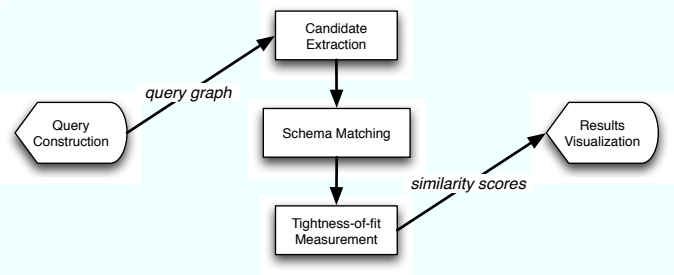


Figure 3: Schema search algorithm data flow

Candidate Extraction

The input query-graph Q is a forest of trees consisting of schema fragments and keywords, as shown in Figure 1. The example illustrates that Q can represent several graphs, where each keyword is represented as a graph of one item. The query-graph abstraction can capture multiple query formats, including relational and XML.

The system contains a document index of the schema corpus, which we build offline. Each schema in the index is represented as a document, for which we store a title, a summary, an ID, and a flattened representation of each element in the schema. Our inverted index stores a term dictionary of frequency data, proximity data, and normalization factors, providing a fast and scalable filter for relevant candidate schemas.

When searching the index online, we first create a list of keywords by flattening the query graph Q and performing keyword matching on the document index. We use a variant of standard TF/IDF to obtain an initial coarse-grain matching. To preserve recall, the candidate extraction algorithm need not match all search terms; rather, match scores are computed independently for each search term and summed to produce a coarse-grain score for returning the top n candidate results. A coordination factor, defined as the number of terms matched

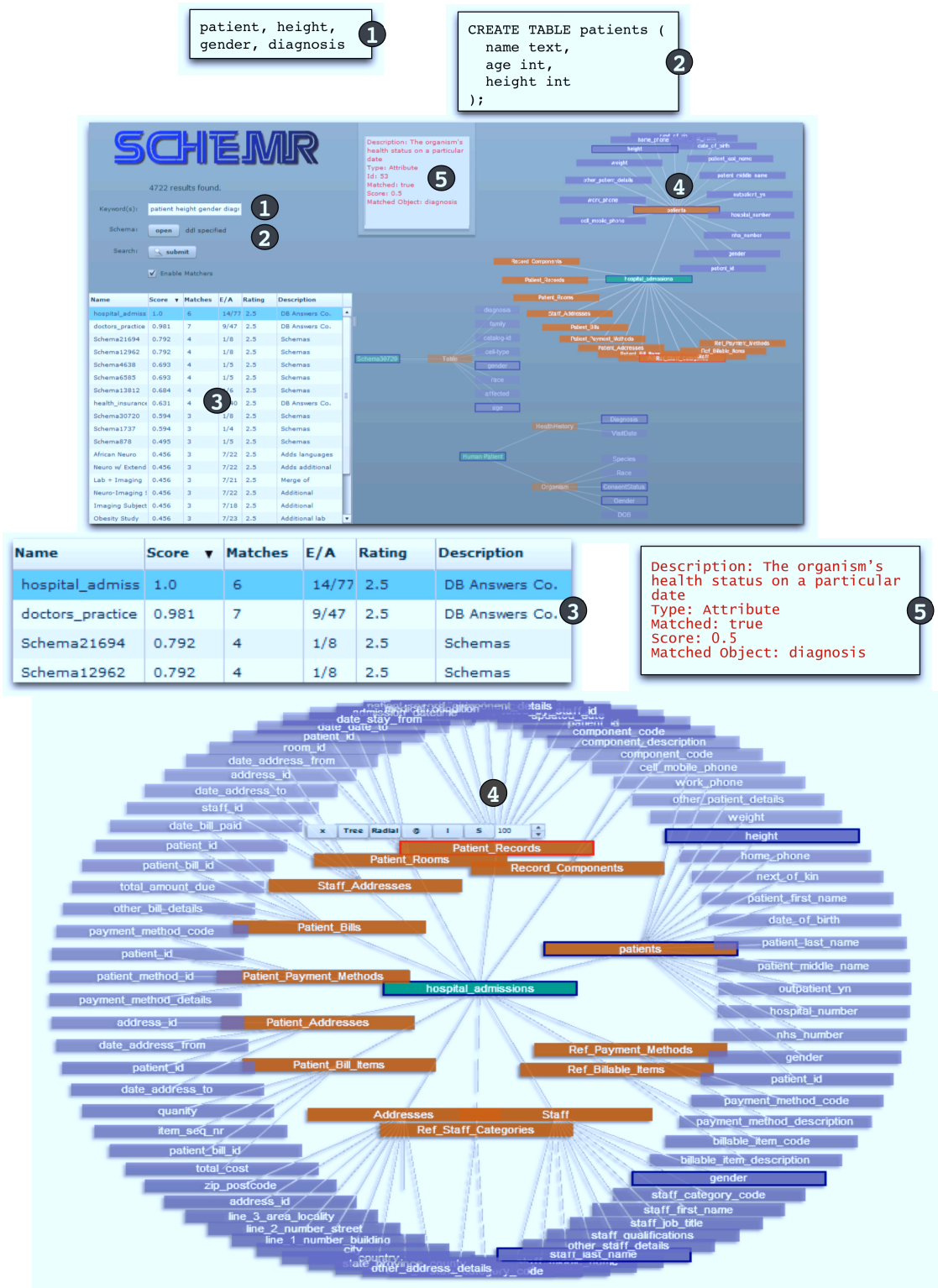


Figure 2: Search results for a keyword + schema fragment query. (1) Search keywords (2) DDL schema fragment specified as part of query (3) Tabular view of search results allows sorting and comparison (4) Schema visualizations allow side-by-side schema comparison. Node color corresponds to schema element types (e.g. entity or attribute). Visualization types include hierarchical tree-view and radial view (shown). Nodes can be collapsed and expanded to allow drill-in on particular schema elements in greater detail.

divided by the number of terms in the query, is multiplied into the coarse-grain score in order to reward results which match the most terms in the original query.

Schema Matching

The top candidate schemas are evaluated against the query-graph and ranked using an ensemble of fine-grained matchers. We summarize two matchers we found to be most useful, but note that other matchers may be used as well.

A *name matcher* normalizes terms and computes n-gram overlap between query terms and terms in the indexed schemas. Each schema element in the query is parsed into a set of all possible n-grams, ranging in length from one character to the length of the word. Each n-gram is then ranked against each element of the candidate schemas to compute a final match score. We found this matcher to be particularly helpful for properly ranking schemas containing abbreviated terms, alternate grammatical forms, and delimiter characters not in the original query.

A *context matcher* builds a set of terms from neighboring elements, and tries to capture matches when neighboring-element sets are similar to each other [6].

Each matcher produces a similarity matrix between query graph elements and schema elements. Each (query element, schema element) pair has a corresponding value which describes the match quality – a value between 0 and 1, For every candidate schema, the similarity matrices of the different matchers are combined into a single matrix containing *total similarity scores*. We combine the scores from each matcher with a weighting scheme, which is initially uniform. As Schemr is utilized in practice, we can record search histories to create a training set of search-term to schema-fragment matches. With such a training set, we may then determine an appropriate weighting scheme. For instance, Madhavan *et al* use a "meta-learner" to compute a logistic regression over a training set of schemas [5].

Tightness-of-fit Measurement

Schemr's task, in this phase, diverges from the traditional aim of schema matching: rather than generating mappings between elements, we use the similarity matrix of total similarity scores to create an overall score that captures the semantic intent of schema search. Our principle here is to measure the tightness-of-fit by minimizing the distance between relevant elements in a result schema.

We begin by selecting the maximum value of each schema element's entry in the matrix as the final match score for that element. Next, we apply penalties to the scores of the schema elements based on a relative distance measure and take the average of the scores to arrive at a final score for the entire schema.

The intuition behind our distance measure is as follows.

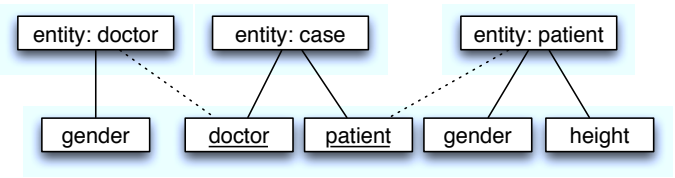


Figure 4: An example schema showing only *matched* schema elements

For elements e_i and $e_j \in E$:

- If they are in the same entity, no penalty.
- If they are in the same entity neighborhood (transitive closure on foreign key), then a small penalty applies.
- If they are in unrelated entities, then a larger penalty applies.

There can be many configurations by which a set of query-graph elements match a set of result schema elements. Each such configuration consists of penalties on elements computed with respect to a particular anchor entity. Given an anchor entity, the scores of elements in other entities are penalized by their distances to the anchor and averaged. This calculation is repeated for all possible anchor entities, and the maximum of all calculations is selected as the final match score for the schema.

Continuing with our original health clinic example, consider the following simplified candidate schema of matched elements in Figure 4. First, `case` is selected as an initial anchor entity. No penalty is applied to the scores of the `case.doctor, case.patient` schema elements, because they reside in the same entity as the anchor, whereas a small transitive closure penalty is applied to the scores of `patient.height, patient.gender, doctor.gender`. Finally, the penalized scores of the schema elements are averaged to produce a score for the `case` anchor. Next, `patient` is selected as an anchor entity. No penalty is applied to `patient.height, patient.gender`, the small transitive closure penalty is applied to the elements in the `case` entity, and a larger penalty is applied to the elements in the unrelated `doctor` entity. Finally, this calculation is repeated with `doctor` as the final anchor entity, and the maximum value of the three anchored calculations is returned as the final match score of the schema.

For a set of similarity scores S, each choice of anchor element A results in penalties P. A tightness-of-fit score t can be computed by $t = \sum(S \cdot P)$. We are interested in the configuration which maximizes the tightness-of-fit score:

$$t_{max} = \arg\max_{A} \sum (S \cdot P_A).$$

We use this total score to rank the final search results.

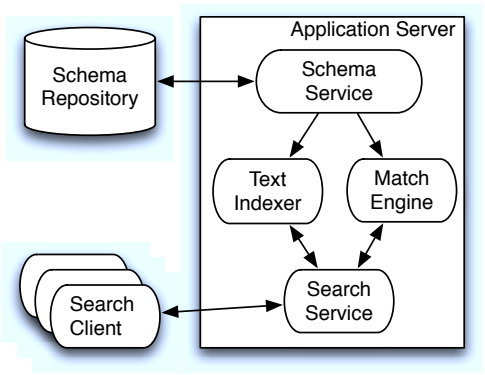


Figure 5: Schemr system architecture diagram

Architecture and Implementation

Schemr's architecture (Figure 5) features a web-based GUI for entering search terms and graphically reviewing search results. The GUI processes a set of search terms and delivers them as a request to the *Search Service*.

On the Schemr server, we use the open-source schema repository Yggdrasil [8]. At scheduled intervals, an offline Lucene [11] *Text Indexer* flattens schemas from the *Schema Repository* to construct or update the document index.

When a request is received by the server, the query is initially flattened into a collection of keywords and used to filter candidate schemas from the document index. These candidate schemas are next passed to the *Match Engine*, where fine-grained matchers are used to compute a final relevance score for ranking the candidate schemas. This list of candidate schemas, along with their corresponding score, is finally sent as an XML response to the client.

When the user clicks on a search result to view the visualization, another request containing the schema ID is sent to the server. The server performs a lookup of this ID in the schema repository and returns a graphical representation of the schema to the client as a GraphML[10] response, which is parsed and displayed on the front-end.

Visualizations

Schemr visualizes result schemas in an interactive GUI, supporting panning, zooming, auto-layout, and drilling-in. Our client is implemented using Adobe Flex and the Flare visualization toolkit. Using Flash ensures cross-browser compatibility without any additional browser-handling code. All search requests and visualizations are dynamically retrieved using asynchronous HTTP requests.

Schemr's user interface features two panels (Figure 2). The left-side search panel allows users to supply a query in the form of a keyword search or a DDL/XSD schema fragment and lists ranked search results in a tabular format. The right-side results panel provides a workspace for users to explore graph visualizations of schemas. In graph visualizations, element nodes are encoded by color, and multiple graphs can freely be compared side-by-side and explored in further depth. Clicking on a graph node displays detailed information about the schema element in a toolbox, and double-clicking on a graph node re-centers the layout of the graph such that the new node is in the center. We allow for multiple graph layouts, including a hierarchical tree layout and a radial layout. To ensure Schemr scales to very large schemas, we cap the displayed graph depth to 3. To drill in on a particular branch at a greater depth, users can simply double click on a node to view its descendants in further detail. To ensure Schemr scales to very large schemas, we plan to employ schema visualization and summarization techniques, such as those proposed in [7, 9].

Applications

Schemr's search capabilities have been tested on a repository of over 30,000 public schemas, both relational and semi-structured, small and large, spanning many domains. These schemas came a collection of 10 million HTML tables [1], and were filtered by removing schemas containing non-alphabetical characters, schemas that only appeared once on the web, and trivial schemas with three or less elements.

We plan to make Schemr available as a part of an open-source information integration framework, OpenII [8]. As a module of OpenII, other framework components enable new schema search applications and scenarios, magnifying Schemr's benefit. For example, integrating Schemr with schema import and export functionality gives users motivation to build metadata repositories. As well, integrating Schemr's search functionality with a codebook that contains data types like units, date/time, and geographic location, would encourage a deeper standardization of data types alongside schema search results. With an OpenII community of users searching the repository, collaboration functionality that provides usage statistics and comments on schemas would improve schema search results. Finally, integrating Schemr with a schema editor would allow for a new model development process, in which search results are iteratively used to augment a schema. In this process, we can also capture implicit semantic mappings between schema elements, information on schema re-use, and the provenance of new schema entities.

3. SUMMARY

Schemr demonstrates an effective approach to schema search and visualization. It uses a novel combination of document based filtering, schema matching, semantics, and structure-aware scoring to efficiently search and visualize large schema repositories. Schemr can be internally deployed as a standalone tool for organizations to search and share schemas, facilitating the schema design process and paving the way for information integration. Additionally, Schemr will play a role as a module of the OpenII framework, serving to improve the accessi-

bility and benefit of many information integration applications.

Schemr can also be deployed as a publicly available web service. To facilitate finding quality schemas in a large public repository, we plan to incorporate collaborative functionality such as mechanisms for users to leave ratings and comments on schemas. Through these comments, users can suggest improvements or additions that can be made to schemas. Ultimately, we hope that this will evolve into a general repository for storing multi-purpose schemas to meet the community's needs. In a sense, we are hoping to democratize development of standards and consequently improve the quality of schemas in the data ecosystem.

Acknowledgments

We are grateful to Peter Mork, Arnie Rosenthal, Len Seligman and Chris Wolf who provided invaluable advice and the Yggdrasil schema repository. We would like to thank Kristin Barker, Harr Chen, Tyson Condie, Joe Hellerstein, Neal Lesh, Jamie Lockwood, Tapan Parikh and Sanjay Unni.

4. REFERENCES

[1] M. Cafarella, A. Halevy, D. Wang, E. Wu, and Y. Zhang. Webtables: Exploring the power of tables on the web. *Proceedings of the VLDB Endowment*, 1(1):538–549, 2008.

[2] K. Chen, J. Madhavan, and A. Halevy. Exploring schema repositories with schemr. In *Proceedings SIGMOD*, pages 1095–1098. ACM, 2009.

[3] A. Doan, P. Domingos, and A. Halevy. Learning to match the schemas of data sources: A multistrategy approach. *Machine Learning*, 50(3), 2003.

[4] M. Franklin, A. Halevy, and D. Maier. From databases to dataspaces: a new abstraction for information management. *ACM Sigmod Record*, 34(4):27–33, 2005.

[5] J. Madhavan, P. Bernstein, A. Doan, and A. Halevy. Corpus-based schema matching. In *Proceedings of ICDE*, pages 57–68. IEEE, 2005.

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

[7] G. G. Robertson, M. P. Czerwinski, and J. E. Churchill. Visualization of mappings between schemas. In *Proceedings SIGCHI conference on Human factors in computing systems*, 2005.

[8] L. Seligman, P. Mork, A. Halevy, K. Smith, M. Carey, K. Chen, D. Burdick, C. Wolf, J. Madhavan, and A. Kannan. Openii: An open source information integration toolkit. In *Proceedings of SIGMOD*, 2010.

[9] C. Yu and H. V. Jagadish. Schema summarization. In *Proceedings VLDB*, 2006.

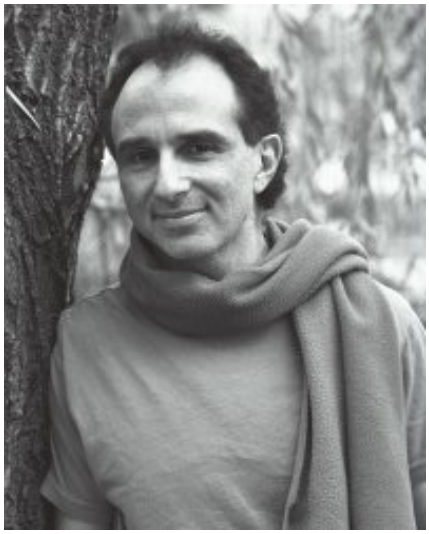
[10] Graphml file format. `http://graphml.graphdrawing.org`.

[11] Lucene. `http://lucene.apache.org`.

Dennis Shasha Speaks Out

on How Puzzles Helped His Career, What Drives Him to Write, How We Can Help Biologists, the Principles Underlying Database Tuning, Why He Wears Shorts All Year, and More

by Marianne Winslett



Dennis Shasha
http://cs.nyu.edu/shasha/

Welcome to ACM SIGMOD Record's series of interviews with distinguished members of the database community. I'm Marianne Winslett, and today we are in Providence, site of the 2009 ACM SIGMOD/PODS Conference. I have here with me Dennis Shasha, who is a Professor of Computer Science at the Courant Institute at New York University. His research interests include biological computing, database tuning, cryptographic file systems, and pattern recognition. He is a prolific writer, both within and outside of computer science.

Let's start off with what our readers really want to know: do you really wear a scarf and shorts every day, through both summer and winter?

Yes, I'm afraid I do. In the winter I get a lot of funny comments, like people asking me, "So, what are you drinking, man?" And one time I gave some money to a guy on the street, and he said, "You know, I should give you money, you should buy some long pants!" So it has been worth it just for the comments!

Have you ever gotten frostbite?

No, actually I am warmer wearing shorts, surprisingly, because the only part of me that gets cold is my hands. Because I wear shorts, I put my hands in my pockets. Because shorts are so thin, I get warmed by my legs, so I am actually warmer in shorts than in long pants.

What about the reverse? In summer when you've got your scarf on, how does that work?

As I said, only a few parts of me that get cold. One part is the neck, and the other part is the hands. So it actually makes sense. Everything about me is a little bit strange, but it makes sense in that context.

Have you ever gone any place really cold during winter?

Waterloo, Canada in February. The waiter came down in the hotel and said, "Sir, you do realize we are in Canada, don't you?" But it was fine.

Biological computing: what's in it for database researchers?

I think it is a great field because computing, after all, started as an aid to science, and we are going back to that tradition when we help biologists. Biologists need a lot of help, in all kinds of areas. Above all, what they care about is experimenter time. They don't care about computer time, per se, as long as they don't need days of computer time. If a program runs for 20 minutes or 30 minutes, it's okay. They do care about reducing the number of experiments they do, and still getting more or less the same results. Also, they want to know how to get stronger results from the experiments they do. I help biologists design experiments using statistics and combinatorial design, so that they don't have to run experiments on every point of an entire search space, and they can examine just a subset of that search space.

Very often, scientists' search space for experiments involves many variables whose values they can modify. I work with plant biologists. They might add more nitrogen or less nitrogen, more carbon or less carbon, and so on. With all these possibilities, the search space can contain millions of points, if you consider all the possibilities. Using combinatorial design, you can reduce it to perhaps a few hundred, and still get a very well sampled experimental space. Then on the analysis side, what is really nice is that you need a little bit of everything. You need some machine learning, data mining, statistics, and there is a lot of simulation. So overall, there is a lot that computer scientists can do to help scientists, and if a database researcher is willing to be all of computer science to the biologists, then there is a lot to be done.

Should we be training the biologists differently? Do they need to know more computer science than they are learning, so they can do it themselves?

I think biologists do quite a bit themselves. For example, all biologists know how to use BLAST for comparing sequences. But on the other hand, there are also things that the computer scientists need to be doing. Those things often have to do with either a new kind of analysis that the biologists wouldn't be able to do by themselves, or just looking at the data differently, or sometimes even helping the biologist design a better experimental program. One biologist that I work with, a woman named Gloria, has been working on a certain kind of network for nitrogen uptake in plants. What she does is great, but what we would really like to do is to take this network and reverse engineer it, like electrical engineers would do.

What do you mean when you say "reverse engineer it"?

We'd like to find out which genes are connected to which other genes. It's as if we were probing an electrical circuit and finding out which circuit elements are connected to which other circuit elements. In doing that, an electrical engineer puts current into a circuit element, or maybe detaches a wire. With plants, we can't exactly detach a wire, but we can make genes overexpress, so that they produce more protein than they would normally, and then we can see the effects of overexpression. By doing that, we can march through the circuit, in principle, and find out exactly what is happening. I find that very exciting. Gloria and her lab have bought into this idea, and they are starting to do those kinds of experiments, so it is a wonderful collaboration.

But don't you get all sorts of side effects once that one gene starts producing too much protein?

Sure, but that is exactly what you want to find out. You find out the side effects, and you can find out which other genes that gene is affecting. So that helps you figure out, "Ah-ha, that gene must be connected to the other gene!"

It sounds like a close partnership that you are talking about.

Yes, it is really a great partnership. We have a meeting every week, and we have cookies, and we make a lot of mistakes in the other field… One time I said "Alabama" when I meant "alanine", because both words have the "ala" prefix and I just forgot. So we really have a good time together.

With Cathy Lazere, you wrote a book of biographies of great computer scientists, called Out of Their Minds: The Lives and Discoveries of 15 Great Computer Scientists. Tell us a story from that book.

When we wrote *Out of Their Minds*, we wanted to write about the seminal thinkers of computer science. We don't mean Alan Turing, of course, so much as the people who have made computing practical in a technological sense. So we chose people who had already won the Turing Award, plus others who won the Turing Award soon after we interviewed them, although I am sure there was no causal connection. We started with algorithms people, like Michael Rabin, Dijkstra, Knuth, Lamport, and Cook and Levin for complexity theory. We interviewed languages people: McCarthy, Alan Kay, and Backus. We interviewed computer architects, and we interviewed people in artificial intelligence. I avoided the database field because I felt that if I had interviewed one person, maybe other people would have been upset. Probably that was a mistake, so perhaps including the database field would have been good too.

A typical story from that book is that of Backus. Backus in some ways personifies a very unusual kind of researcher, who nevertheless has the emotive influences that drive many researchers that I know. Mainly, he was easily angered. He had a very mixed, not very studious, childhood. At 25, he was graduating from the general studies program at Columbia, with a math degree, not knowing what he wanted to do. He got a job at IBM almost by accident: he was taking a tour of IBM, and somebody said, "You can interview here if you want. You can maybe get a job." He took a test and they hired him.

At IBM, Backus was programming in machine language, which he found horrible. He started what became the predecessor of FORTRAN, namely, a way of doing floating-point arithmetic. When he proposed FORTRAN, he said, "I am doing it because I find programming is too tedious, too difficult for people." Interestingly enough, he was opposed in this desire by John von Neumann, who thought that programming really wasn't that hard. But von Neumann was not your average person, and Backus understood this, and the IBM management understood it too. So when Backus started the FORTRAN project, he was able to get the support of the IBM management, who understood that if they could make software programming easier, then people would buy more computers. And that is what happened.

Later, Backus participated in the ALGOL discussions. He said it was very frustrating; people would just give an example for this, give an example for that, and nothing was getting anywhere. And so he developed Backus-Naur form, which is really very similar to the Chomsky notation for context-free grammars, just to make things clearer. So just because he was annoyed, he did a lot of his inventing.

You make it sound like he is an accidental computer scientist.

I think he is not accidental, because I think a lot of invention is annoyance. It is true that he might have gone into another field, because he wasn't a particularly outstanding math student, by any means. So he is an accidental computer scientist in that sense.

I have a colleague, David Mazières, who is at Stanford now, and he is also very motivated by anger. It is a fun anger, it is kind of enjoyable to watch. He gets angry, and then he does something about it. He builds a better system.

What angers you?

I am not one of those kinds of people. I like hard problems where I feel that a solution will really change things. Of course, it doesn't always work out that way, but sometimes it does.

Have you had a favorite hard problem in the past?

I have had a couple that I really enjoyed. One was the work that we did in tree matching. This work got started because a student of mine, Kaizhong Zhang, came to me and said, "You know, this journal paper claims to have a really fast algorithm for this problem, but the algorithm doesn't actually work." We looked at it together, and indeed, it didn't work. Sometimes a problem seems closed because somebody else claims to have solved it, but in fact the problem is not really closed because it has not actually been solved. Noticing this discrepancy can be a fun way to start a research project.

A sequence of accidental coincidences led me from one research area to the other: meeting people, understanding their problems, coming up with algorithms to solve them, and getting the algorithms used. That project on tree matching led to other projects on graph matching, and led eventually to my work with biologists, because one of the applications was RNA secondary structure.

You are a puzzle column writer for Dr. Dobbs' Journal *and for* Scientific American. *What's that like?*

Dr. Dobbs' Journal stopped publishing a couple of years ago, and *Scientific American* is going through economic problems, so it is not clear that the column is going to continue. But it has been fantastically fun. As the column writer, you write puzzles and people send in solutions. And when people send in solutions, you become friends with them. I have what I call my "puzzle brain trust": sometimes I invent a puzzle that I can't solve very well, so I send it out to various members of the puzzle brain trust, and sometimes they come up with much better solutions than mine. It has been a remarkable experience, because I haven't met my puzzle brain trust in person. In fact, I just discovered a couple of years ago that one of them happens to be deaf, and I had no idea. It has been just a wonderful experience in terms of the intellectual back and forth with many of the people I have communicated with. Nothing delights me more than finding somebody who has a better solution than I do.

The puzzle column also has helped me formulate problems for the biologists that I work with. Sometimes the discipline of making a puzzle out of a problem can help you simplify the problem to its essentials. When you do that, you really start to understand it, and then you start to ask the right questions, and then sometimes you find a better solution. For example, the combinatorial design that I use for biologists turned into a safe-cracking problem as a puzzle. Instead of a dial on a safe, you have many three way switches, and the goal is to open the safe, given that it will open only if two switches (and you don't know

which ones) are in a certain configuration. It's a combinatorial design problem, and when you explain it that way to biologists, they understand it – possibly mainly because they secretly want to break into safes!

When you publish these puzzles, do you always already have a solution?

I already have a solution, but sometimes it is not the best solution. Many of these problems are continuous or NP complete, and so it is difficult.

I did one puzzle about Zambonis, which are machines that clear off the ice after people have been ice skating on it for a while. My inspiration came from observing the Zambonis and noticing that although the drivers seemed to be having fun, they definitely were not using the most efficient route. About a week ago, I received an email from the Zamboni people, who said, "We would definitely like to find the solutions to this puzzle."

In the problem, you have an ice rink, abstracted to a bunch of points filling an oval shape. The Zamboni machine has to pass over every point, but cannot turn at more than a 45 degree angle. How do you make it go through all those points in as short a time as possible, so that people can start skating again? I suppose with enough computing power you could solve that problem, even though it is quite complicated because you have the 45 degree constraint. Anyway, some readers found a really nice solution. Another reader found a really beautiful solution for two Zambonis that was also kind of important, because the two Zambonis can work together without colliding. So we will see, maybe Zamboni will use it!

You've also written about recent Russian immigrants. What is the story there?

I like to do things that interest me, and I don't sleep much at night. I usually sleep 5 or 5 ½ hours, and then I take a few 10 minute naps during the day. When I wake up in the middle of the night, often I want to write, because I just like it. And sometimes daily experiences influence me.

In the case of the Russians, I was director of undergraduate studies in the early 1990s, when a lot of Russians were coming to New York. They were kind of bimodal. Some of them were just brilliant. Some of them thought they were brilliant, but weren't. There weren't many in between, for some reason.

One day a young man came into my office with his teacher, and he was a *very* young man, just 12 years old. His teacher said, "He's in my freshman Pascal class, and there is a problem." I asked what the problem was, thinking that the student was too young and couldn't understand the material. But the teacher said, "Viktor is too good." I said, "Oh, really, so tell me Viktor, what have you done? What is the biggest program that you've written?" Viktor said, "Well, I've done a simulator for high temperature superconductors." And I said, "Oh, that's good."

My colleague asked Viktor about the math he used, and Viktor was explaining Newton gradient methods and all those sort of things, and he certainly understood what he was doing. So then I said, "Okay, Viktor, which programming languages do you know?" "Pascal, FORTRAN, and Lisp, and C, and C++, and I think that is all," he said. "Do you know any graph theory?" "A little bit," he said. "Could you write us a connected components algorithm?" "In which language?" he asked. So I said, "Oh, I don't know, maybe a set oriented language." So he does that in his 12 year old handwriting, from the top of the board to the bottom, without a cross-out, and it is perfect.

My colleague said, "You have a loop there, and you have to compare two sets in that loop, that's an expensive operation. What will you do?" "Well, I will probably use bit vectors," Viktor said. Then my colleague said, "You could also just have a flag that changes its value if there is some change in the loop." And Viktor said, "Oh yes, but it wouldn't be so pretty."

I felt that I had to write a book about these people. I wrote with a Russian playwright, and we wrote about mathematicians and scientists but also business people and sex workers and writers. So it was quite an eclectic crew, and it was really a fun book to write.

When I write, I try to write something nobody else is writing. I really respect people who write wonderful textbooks, like let's say, a real database textbook, but I would never try to do that, because I feel that they are already doing a great job. Even if conceivably I could do a better job, it would only be marginally better. For example, I wrote my book about database tuning because there was no such book. There were books about database tuning for particular systems, but never the general principles.

Are there general principles? Can you generalize beyond what each separate system does?

I think there are a lot of general principles. One principle is that people often tend to tune things that they notice are bad, even if they are not important. A second principle is that very often there are systematic problems because people don't realize that one fundamental principle of all databases, in fact all of computing, is that starting something is expensive, but once it's going, it is quite cheap to have it continue. For example, reading a sector of a block is almost as expensive as reading the whole block. Emitting an SQL query every time you go through a Java code loop is much worse than having just one SQL query and then stepping through the resulting data. That theme occurs again and again, and I call it "starting is expensive, but running is cheap."

Another principle is that partitioning is not understood too well. People understand partitioning data across pieces of hardware, but they don't really understand that there is also temporal partitioning. Very often, you might want to do some work during the day, and other work during the nighttime. When you say that, people respond, "Well, of course," but it's not the first thing that comes to people's minds. For example, banks send out statements. When they used to send out paper statements, they would send 1/20 of the statements every day. And that makes sense. It doesn't matter that you don't get your statement at the end of the month – it could be on the 12th, but who cares as long as you get it every month? And similarly, for production databases, one can take advantage of temporal partitioning.

Those are three of the principles. Surprisingly, those principles are almost generative. You can apply them to almost every level of the database, from buffer management to application design.

Do you think that the general principles of database tuning would fit into a standard CS curriculum?

I think so, but of course I am very biased. I teach it, and students love it because it is very practical. I think we do a very good job in this community teaching a database course on SQL and normalization. Once we get to indexes and transaction processing, we have to recognize that most of our students are barely going to be conscious of that once they leave – indexing perhaps, but not how to build the index, just what indexes do. Although it is all very valuable, we should recognize that database tuning is what they will actually do if they use databases, either as database administrators, or even as sophisticated application programmers. Because how many will work at Oracle, Microsoft, or a few other places?

We tend to make our students take a course on operating systems, but most of them don't go on to design operating systems. Does that represent a failure of our educational system?

No, because operating systems and database internals give people a good sense of what system building is about, and that is important. Very often, students go on to write concurrent systems, so they have to understand system building, and operating systems and transaction processing are great ways to learn that. But, nevertheless, I think there is a place for database tuning, at least a little bit. I welcome anybody in the community to take advantage of my notes on database tuning, which are widely available. Philippe Bonnet has a wonderful infrastructure for a database tuning experiment, and that is also available. It uses mySQL, and we have some scenarios about a travel application, and some other scenarios.

Tell us about your life as a fiction author.

I also loved puzzles, because puzzles helped me overcome my problems in my first job, which was to design circuits for large mainframe processors at IBM. Although I technically had an engineering degree, really I'd only studied information theory. So when I got to IBM, I didn't really understand electricity, not really. I bought myself a Heathkit and I built stuff, and I really tried to understand what was going on. But what really helped was reducing what I had to do to puzzles. Then through those puzzles, I tried to understand more details afterwards.

One of my jobs was to design circuits to check other circuits. The idea was that if a circuit failed, another circuit could say that that circuit failed. Circuit elements were still expensive enough at that time that we didn't want to completely duplicate the circuit. We wanted something that was economical, yet still could check the other circuit. Take a decoder circuit for example, a 4-16 decoder. The output should have only one element that's set to 1, i.e., only one of the 16 lines should be a 1. How do you do that in just a few gates? That puzzle appears in the very first of my puzzle books. The naive solution is to take every pair of the 16 and see whether both outputs are 1, in which case there is a problem. But that is many, many, many tests, at 16 choose 2 circuits. One can design a much better and more efficient circuit by noticing that this is 4 bits, so if two of these lines are 1, then they must differ in one bit.

So puzzles always helped me. And then I wondered, how can I give puzzles to other people? I always really loved Sherlock Holmes, except that Sherlock Holmes periodically would just leave and then come back and say that he had discovered something, which would often be the key to the whole mystery. I felt that was unfair to the reader. Any reader who tried to solve the mystery himself/herself was robbed of that privilege if Sherlock Holmes pulled a wildcard out of his pocket, so I said, *no wildcards*.

Dr. Ecco is kind of like Sherlock Holmes. He has a friend named Prof. Scarlett, and Prof. Scarlett acts the Watson role. Prof. Scarlett explains the puzzle to us, but never does the reader get less information than Dr. Ecco. So if Dr. Ecco can solve it, the reader can too. And indeed, readers do solve it. Most of my puzzle books have puzzle contests where the solutions aren't given in the back, and the person who wins that contest gets free tickets for two to London and back from New York. It has worked out well.

When you read a murder mystery, do you usually figure out who did it before the end?

I try. Sometimes it doesn't work. But I like to, I like that idea.

Why is it that modern mysteries have to be murder mysteries? Why can't they just be mystery *mysteries?*

I agree with you. I think there is plenty that is mysterious and doesn't have to involve death. *The Name of the Rose*, for example – Dr. Ecco's name has two Cs, and Umberto Eco only has one C, but nevertheless, *The Name of the Rose* was a very inspiring book for me. That mystery fundamentally didn't need to involve death. And that book really involved the reader, took the reader to an entirely different world.

What will your next book be about?

Cathy Lazere and I wrote a book that's at the publisher's now. It's called *Natural Computing: DNA, Quantum Bits, and the Future of Smart Machines*. [It came out in May 2010 in English, May 2011 in French, and soon in Japanese.] This book is much more speculative than *Out of their Minds,* but in the same format: biographies and what people are doing now. We look at future directions in computing based on people who are trying to solve extremely hard problems. For example, how do you make spacecraft survive for a hundred years if they are going to make a long trip? Another typical hard problem is protein folding. Another is how to make robots that fend for themselves. There are many very difficult problems that I think require a new kind of computing, and the new kind of computing we seem to be seeing is based on a life analogy. Either it involves genetic algorithms, or more generally it involves feedback in very interesting ways, so that systems are built to be adaptable. Very often, not only do they involve nature in a metaphorical way, but also in a real way. For example, we interviewed Jonathan Mills from Indiana, who does analog computing, which people thought was completely dead, but which can be really good for certain things. This summer, in fact, we are doing an extremely intensive project with him on protein folding. After all, proteins fold in a millisecond, but the best computers for doing this, like David Shaw's Anton, take months to fold a small protein. And those are multi-million dollar machines! There is something wrong with that picture, because clearly, the protein is not computing like Anton does. So it is quite possible that there are other paradigms that could work better.

Beyond the puzzle aspects, are there other characteristics of the types of problems that you like?

I am really interested in very challenging problems in any part of computer science. I think there is going to be a lot more autonomy in computing systems. Even in sensor networks there has to be a lot more autonomy, because you can't possibly go out and repair all these sensors. It is even more so in some of the wild things that people are doing, like in DNA computing, where there are literally billions of little DNAs running around annealing with another, or even in cellular computing where on your thumb you have more cells, little pieces of bacteria, than all the people who live on the planet. It is just an enormous question of scale. There are going to be interesting problems where you don't have control of the computing elements: how can you nevertheless solve big problems? These problems will involve data in as much as each of these computing elements will have data. I think challenges like that will be entirely new and possibly give entirely fresh perspectives on all aspects of computer science, including databases.

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

I have enjoyed so many things that I have done. I really like that some of the work has surprising applications. We have gotten requests for the tree matching software from sociologists, linguists, all kinds of people. The time series work that we've done has been downloaded by the usual suspects on Wall Street, but also has been used by people who are interested in "query by humming", in music, and also in astrophysics! So sometimes, the work has found a new life in unexpected ways. I have really enjoyed large data puzzles, where there is a lot of data, but also some combinatorial structure to make use

of. I don't really like problems without a combinatorial structure, that are just large for large's sake; I guess nobody does. If there is a nice puzzle, something clever that we can do, then I really like it.

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

The advice I have is almost the same as the advice that Dijkstra gave when we interviewed him for *Out of Their Minds*. He said, "Choose problems that are at the limit of your ability, not too hard, but not too easy either." Before you have tenure, you have to publish, but nevertheless, it is important to realize that you want to publish things that people will appreciate, that people will say, "Yeah, this was clever, and I couldn't have done that," or, "If I could have done it, I would have been really proud of it." My other advice is to go out in industry, or government, or wherever, and find people who have real problems. When the founder of my institute, Richard Courant, came to America, he would take his students to large engineering organizations and tell them to talk to the engineers. Not to the physicists and not to the mathematicians there, but to the engineers, because they knew what the real problems were. I really believe in that advice because it has led me to database tuning, to the time series work, to work with the biologists and write a funny little book called *Statistics Is Easy* – every time I have done something it is because something real I have seen has motivated me to do it. Sometimes it is not research in the hardest sense, e.g., *Statistics Is Easy* is just an easy introduction to resampling statistics. But it is nevertheless useful because most computer scientists don't like statistics. I think utility is a great motivator, and I think it is sometimes underestimated in academia.

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

That's a good question. I am curious about many things, and sometimes that leads me to try to do too much. I am working with biologists, I am doing a DNA computing project; I think it is great for students, because they gain a lot from it, but sometimes it doesn't end in, let's say, a publication. I'm not really too worried about that, but I think that would be the one thing I would change: I would maybe say *no* more.

How do the students get jobs if the work does not lead to publications? Even if you have tenure, your students will be looking for jobs, and if they don't have the publications they can't get the jobs.

I quickly evaluate whether a student is likely to go to academia. Many students don't really want to. If the student does, then we concentrate more on publications. Of course, sometimes I am wrong. I had a wonderful student named Yunyue Zhu who won a best paper award in time series, beautiful publication record, and then he went off to Wall Street because the money was so good! So what can I do?

Is he still on Wall Street?

Yeah! You see the headlines about Wall Street, and you think the world is coming to an end. But there were a lot of people on Wall Street who did extremely well, and all my students there are completely employed. It is a very funny difference between the headlines and what's actually going on.

Thank you very much for talking with me today!

It's a pleasure, thank you so much, Marianne.

Affiliation analysis of database publications

David Aumüller, Erhard Rahm
University of Leipzig, Leipzig, Germany
{aumueller, rahm}@informatik.uni-leipzig.de

ABSTRACT

We analyze the author affiliations of database publications to determine the main institutions contributing research results in our field. We consider the publications of the last decade (2000—2009) that appeared in the top conferences SIGMOD and VLDB and in the VLDBJ and TODS journals. We determine the top affiliations in terms of number of papers and aggregate the numbers at the levels of entire countries and continents. Further, we analyze to which degree authors from different affiliations and countries cooperate on jointly authored papers, and study the development over time. We also consider the number and size of affiliations of different countries.

1. INTRODUCTION

Previous bibliographic studies of computer science and database publications mainly focused on the number of papers and citations per author or per venue as well as co-authorship relations [4, 6-9]. However, there has been very little analysis of the affiliation of authors to determine where research results are produced. Commercial bibliography services such as Elsevier Scopus and Thomson Web of Science provide some affiliation information but are still mainly limited to journals. By missing most conferences they do not sufficiently cover the computer science research literature. In [8] the affiliations of database publications have already been evaluated but based on a largely manual effort. The study only considered first author affiliations of publications with more than 20 citations at the time of the evaluation.

In this paper we present a more comprehensive affiliation analysis (considering all authors) based on a largely automatic determination of author affiliations. Determining the affiliation information automatically is quite challenging and requires a substantial effort for information extraction, data cleaning and matching heterogeneous representations of the same affiliations. We mainly extracted the affiliation data from bibliographic portal web sites, such as ACM Digital Library and SpringerLink; in some cases we had to extract the information from lists of accepted papers or directly from the fulltext documents. For conferences, we also determined the type of paper (research, industry, demo) which required the integration of further information such as the tables of contents. The collected affiliation strings are highly heterogeneous (frequent use of acronyms and abbreviations, etc.), often inconsistent and partially incomplete (e.g., "Microsoft Research" without city information). Consider for instance the two strings "MIT" and "Department of Mechanical Engineering, Massachusetts Inst. of Technology, MA 02139, Cambridge, USA" referring to the same institution (neglecting department). The pursued approach for entity recognition and affiliation matching (utilizing existing web search engines) is described in [1]. We extract institution and location information from the affiliation strings but ignore departmental information as this information is unstable over time and not always given. Our affiliation information is thus at the level of institution and city from where it can be aggregated at coarser geographic levels such as country and continent.

Our affiliation analysis focuses on database publications of two top database conferences (ACM SIGMOD, VLDB) and two top journals (ACM TODS, VLDB Journal) over ten years (2000 until 2009). These venues are known to be highly selective and of high quality so that (frequent) publications in these venues can be viewed as a quality indicator not only for authors but also their institutes. In this initial study, we will analyze the number of papers of different affiliations and their countries and continents, and study the development over time. We also analyze to which degree authors from different affiliations and countries cooperate on jointly authored papers. Furthermore, we evaluate the number and size of affiliations of different countries. Due to space constraints, we leave an affiliation-based citation analysis for future work. At *dbs.uni-leipzig.de/affiliations* we set up a website to browse the papers.

In the next section, we provide some base statistics on the considered publications. In section 3 we study author affiliations at the levels of continents and countries while section 4 focuses on the top affiliations and the most prolific authors within.

2. BASE DATA

Table 1 provides some base statistics for the considered papers in the four venues (TODS, VLDBJ, SIGMOD, VLDB). It shows the number of papers that appeared in the two journals and the two conference series. The conference papers are further discriminated by track into research, industrial, and demo papers. In the lower part of the table we differentiate the paper counts by first and second five year spans; we will use these two time intervals to illustrate some temporal trends.

In total we analyze the author affiliations for more than 2,700 papers: over 1,900 research papers and more than 800 demo and industrial papers. Slightly more than a quarter of the research publications appeared in the two journals (per year about 50 on average vs. 140 research papers in the two conferences). The number of papers per year almost doubled during the decade (188 in 2001 vs. 352 in 2009); about 60% of the papers appeared in the second half of the decade.

Year	pubs	Jrn.	Conf.	*Conf. track: r, i, d*			res.
2000	188	26	162	*95*	*31*	*36*	121
2001	203	35	168	*103*	*28*	*37*	138
2002	212	32	180	*111*	*32*	*37*	143
2003	225	35	190	*128*	*17*	*45*	163
2004	292	42	250	*150*	*43*	*57*	192
2005	295	54	241	*150*	*38*	*53*	204
2006	276	56	220	*141*	*26*	*53*	197
2007	318	52	266	*175*	*27*	*64*	227
2008	360	91	269	*179*	*30*	*60*	270
2009	352	83	269	*171*	*39*	*59*	254
1^{st} 5y	1,120	170	950	*587*	*151*	*212*	757
2^{nd} 5y	1,596	331	1,265	*816*	*160*	*289*	1,147
Decade	2,716	501	2,215	*1,403*	*311*	*501*	1,904

Table 1: Base data per year and 5-year periods

Most publications have more than one author so that it is of interest to what degree authors of different affiliations and countries publish together. Fig. 1 provides some base information in this respect by illustrating the relative shares of publications with specific numbers of authors, affiliations, and countries. We observe that less than 5% of all demo and research papers and 25% of the industrial papers are written by a single author; the majority of research publications share two to four authors. While the majority of industrial and demo papers originate from a single affiliation, there are slightly more research papers from two or more affiliations than from a single institution. Almost 25% of the research papers have authors from two or more countries. Table 2 lists the average number of authors, affiliations, countries, and continents per paper. Interestingly, research papers involve on average more affiliations and countries per paper despite a lower number of authors than industrial and demo papers.

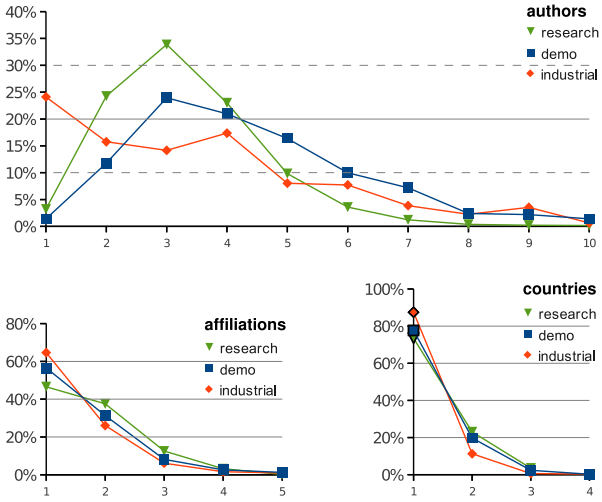


Figure 1: Frequency distribution of publications with distinct number of authors, affiliations, or countries

Entity/track	research	industrial	demo
Author	3.34	3.68	4.58
Affiliation	1.73	1.51	1.62
Country	1.30	1.15	1.25
Continent	1.21	1.08	1.15

Table 2: Average participants per paper by track

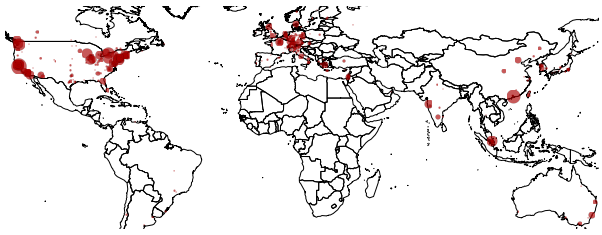


Figure 2: Geographic distribution of affiliations publishing in the top database venues in the last decade

The map in Fig. 2 pinpoints places of the world with papers in our collection. Larger bubbles denote more papers, visualizing the concentration of larger paper counts to only few hubs (US West and East Coast/Great Lakes, Central Europe, Hong Kong, and Singapore).

3. CONTINENTS AND COUNTRIES

We first analyze which continents and countries published most papers in the considered top venues and how the productivity developed during the decade. The reported number of papers per entity, e.g. author, affiliation (institution, city), country or continent, is derived by crediting each entity once when at least one author of the paper is affiliated with it. Thus, the sum of papers can be larger than the sum of unique papers due to multiple authors. In the following tables 3 to 7 we list this total number of papers as *pubs*. We also report fractional counts (*frac)* where each author (and her affiliation, country or continent) is credited only the n-th part of a paper in case of n authors [2, 5]. This is a simple approach to account for cooperative efforts since the fractional counts sum up to the total number of papers. Furthermore, as rough indicators for the degree of cooperation, the average number of contributing entities (authors, affiliations, countries, continents) per paper are shown in the tables. We also list the number of affiliations (institutions at the city level) that contributed publications.

3.1 Continents

We aggregate author affiliations into countries as well as into the continents North America (N.A.), Europe, and Asia. We further aggregate Africa, Oceania, and South America into the Southern Hemisphere (S.H.) as only few papers originate from this region. We observe from Table 3 that by far most papers originate from affiliations in North America (USA and Canada). Almost three quarters of the research papers as well as the industry/demo papers have at least one author from this area. Europe contributes the second most papers followed by Asia. The number of contributing affiliations is somewhat differently distributed since less than 50% (283 of 623) are located in North America. The number of affiliations is relatively high for Europe as we will cover further when discussing the country and affiliation statistics. The level of cooperations (average number of continents per paper) is higher for continents with fewer papers indicating that they depend most on cooperations with affiliations from other continents to publish in the top venues.

Cont.	affil	pubs	*frac*	res, ind, dem	C./p (r i d)
N.A.	283	**1,982**	*1,766*	1,396, 258, 328	1.3, 1.1, 1.2
Europe	217	**642**	*504*	436, 46, 160	1.4, 1.3, 1.3
Asia	95	**513**	*309*	407, 29, 77	1.5, 1.4, 1.4
S.H.	28	**79**	*51*	63, 3, 13	1.8, 2.0, 1.8

Table 3: Publication counts per continent

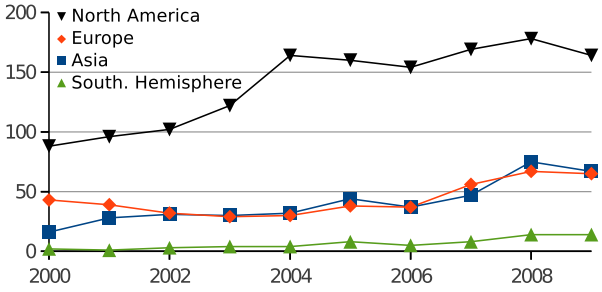


Figure 3: Research paper trends by continent

Fig. 3 illustrates how the number of research papers per continent developed during the decade. While the number of papers increased for all continents during the decade, the strongest increases are observed for Asia and North America. Asia caught up with Europe in the last years and the lead of North America has even increased during the decade. The trends for industrial and demo papers are similar, albeit Europe has retained a lead over Asia so far.

3.2 Countries

Table 4 lists top countries contributing most publications in the three categories research, industry, and demo. We observe that USA leads by far for all three paper categories. Germany and Canada are runners up for industrial and demo papers, and also among the top countries for research papers. The fractional paper counts lead to a largely similar ordering of countries but can be better used to determine the relative paper shares per country (since the fractional counts sum up to the total number of papers). For instance, the fractional numbers show that US authors contribute about 60%, 75% and 53% of all research, industry and demo papers, respectively. This underlines that the US dominance is especially pronounced for industrial papers as the major DBMS vendors are from the US. The table also differentiates the number of papers in the first and second half of the decade illustrating some interesting trends. Regarding research publications, China and Singapore more than tripled their number of papers in the second half of the decade. For the whole decade, this helped China and Singapore to contribute the second and fifth most research papers from all countries. The dominance of US institutions has slightly reduced since the share of research papers with an US-based co-author changed from 72% in the first to 67% (770/1147) in the second half of the decade. Also, the UK and Australia have achieved significant increases in the second half of the decade, positioning them among top ten.

country	affils	pubs	*frac*	cntr/p	1^{st}	2^{nd}
USA	179	**1,315**	*1,131*	1.32	545	770
China	24	**176**	*124*	1.73	37	139
Canada	15	**160**	*93*	1.89	61	99
Germany	50	**147**	*109*	1.53	69	78
Singapore	4	**102**	*64*	1.92	24	78
Italy	23	**57**	*36*	1.70	22	35
France	24	**56**	*37*	1.79	32	24
India	14	**56**	*38*	1.63	30	26
UK	10	**46**	*29*	1.83	10	36
Australia	14	**46**	*31*	1.76	5	41
USA	115	**247**	*231*	1.15	118	129
Germany	22	**23**	*17*	1.48	11	12
Canada	10	**23**	*16*	1.61	14	9
India	11	**13**	*9*	1.62	7	6
South Korea	6	**6**	*5*	1.00	1	5
USA	116	**305**	*267*	1.28	124	181
Germany	35	**73**	*58*	1.38	28	45
Canada	11	**45**	*27*	1.80	20	25
China	19	**32**	*23*	1.78	12	20
Italy	16	**28**	*20*	1.63	13	15

Table 4: Countries by research, industrial, demo

The average number of countries per paper in Table 4 is especially high for Canada and Singapore, denoting an over-average amount of cooperations with authors from one or more other countries. The high degree of cooperation also led to significantly reduced fractional paper counts for these countries. To gain additional insight, we illustrate in Fig. 4 the number of intra- and cross-country papers for the top five countries. It shows the amount of papers attributed to a single country and between countries, giving overall as well as research, demo, and industrial counts. We observe that cross country collaboration mostly takes place in connection with the USA, especially regarding neighboring Canada, where nearly as many research papers were co-authored with US-based authors as without. Singapore authors co-published to a similar degree with authors from USA and China. Both Germany and China published most papers with co-authors from their own country and co-published a similar number of papers with colleagues from US institutions. While we cannot derive a strong connection between the degree of collaboration and productivity (similar as in [3]), it seems that countries like Singapore and Canada having fewer affiliations than China and Germany were able to substantially benefit from the higher degree of international collaboration.

As shown in Table 4 there are significant differences in the number of contributing affiliations per country even for countries with a comparable number of papers. For instance, Germany has many more research affiliations (50) than China, Canada, and especially Singapore (4). To further analyze this ob-

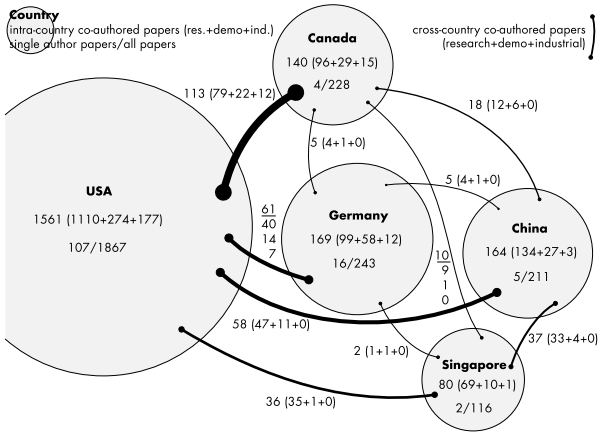


Figure 4: Intra- and cross-country co-operations

servation we also determine the number of authors per affiliation (within the considered ten years) as an indicator of the affiliation size. In Fig. 5 we illustrate for every country its number of affiliations and the average affiliation size; the bubble size indicates the number of papers of the country. We observe that from the five leading countries Germany has the smallest average number of authors per affiliation (6) while Singapore has the largest average group size of 22; the average size of US institutions is twice as high as for Germany. We conclude that large teams with many authors are generally favorable to achieve a high number of papers in the considered quality venues. This will also be confirmed in the next section indicating that the size of the top affiliations is significantly above the country averages.

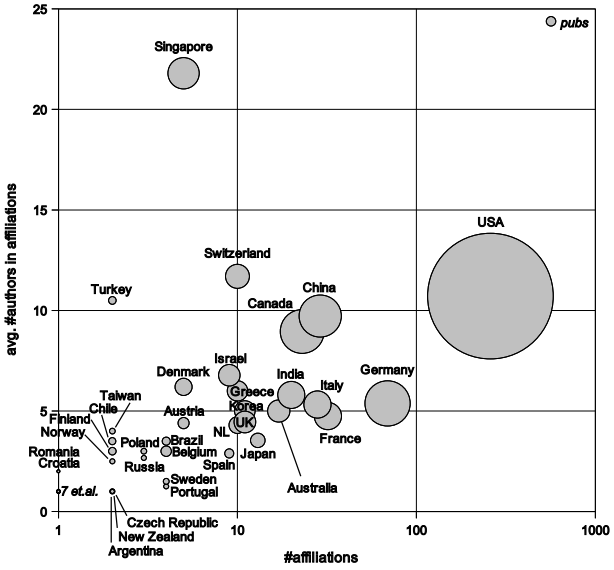


Figure 5: Country publications by no. of affiliations vs. average no. of authors within affiliations

4. AFFILIATIONS AND AUTHORS

4.1 Institutions

Table 5 lists the affiliations contributing most publications in the research, industry, and demo categories. We consider all departments of an institution or company located in the same city as one affiliation resulting in several affiliations for companies such as IBM or Microsoft. The table shows that three US companies with large research departments contribute most research papers in the last decade: IBM San Jose, Microsoft Redmond, and AT&T Florham Park. IBM and Microsoft are also top contributors for industrial and demo papers while DBMS vendor Oracle is only prominently visible for industrial papers. Microsoft Research nearly doubled the number of research papers in the second half of the decade and outnumbered its competitors in this time period.

Regarding research publications from academia we observe that universities from Singapore and Hong Kong have achieved similarly high publication counts as the traditionally strong US universities Stanford, Wisconsin, and Berkeley. While Berkeley and especially Stanford had declining publication counts in the second half of the decade, the National University of Singapore (NUS) and Hong Kong UST had strong increases. Further affiliations with a strongly growing number of research papers include the Canadian universities of Toronto and Waterloo, and the Chinese University of Hong Kong. When aggregating the paper counts per institution across all locations, most research papers come from IBM followed by Microsoft (196 and 128 papers). Aggregating at the city level (across all institutions) reveals that most research publications (133) originate from Hong Kong.

The last column in Table 5 indicates the number of authors contributing to the affiliations' publications. In general the teams of the listed affiliations are very large. With the exception of AT&T, the top ten research affiliations have teams of about 50 or more active authors; the by far largest number of authors (126) comes from IBM San Jose. AT&T has the highest average number of institutions per paper indicating a strong degree of collaboration with other affiliations. Such intensive collaborations seem to facilitate a high number of publications with a moderate number of local authors.

Despite the high number of research papers from Europe (cf. Table 3), no European institution is among the top 20 listed in Table 5. A likely reason for this is the low average affiliation size that we observe for most European countries in Fig. 5. Table 6 shows which European institutions contributed most research papers. We observe that only two institutions had more than 20 active authors so that – with the exceptions of ETH Zurich and INRIA, the team sizes of the leading European database affiliations are substantially below the ones of the globally leading affiliations.

Institution	pubs	*frac*	inst/p	1^{st}	2^{nd}	aut
IBM, San Jose	115	*73*	2.04	49	66	126
Microsoft, Redmond	110	*67*	1.97	39	71	52
AT&T, Florham Park	87	*43*	2.44	49	38	35
Natl. Univ. of Singapore	83	*55*	1.98	23	60	74
Univ. of Wisconsin, Madison	81	*55*	1.88	38	43	71
Stanford University	81	*52*	1.80	54	27	66
Hong Kong Univ. of Science and Technology	79	*46*	2.29	27	52	49
Univ. of California, Berkeley	67	*45*	1.85	37	30	65
Univ. of Illinois, Urbana-Champaign	65	*43*	1.98	25	40	49
Univ. of Maryland, College Park	58	*41*	1.81	30	28	49
Univ. of Toronto	58	*32*	2.29	15	43	37
Univ. of Washington Seattle	57	*41*	1.82	30	27	38
Bell, Murray Hill	55	*32*	2.13	41	14	35
Univ. of Michigan Ann Arbor	49	*31*	2.02	22	27	32
CMU, Pittsburgh	47	*28*	2.19	23	24	48
Purdue Univ., West Lafayette	41	*26*	2.02	15	26	39
Chinese Univ. of Hong Kong	41	*23*	2.32	6	35	32
IBM, Yorktown Heights	39	*19*	2.21	15	24	31
Univ. of Waterloo	37	*22*	1.89	9	28	26
UC Riverside	37	*19*	2.38	13	24	31
Oracle, Redwood Shores	42	*33*	1.67	13	29	88
Microsoft, Redmond	35	*31*	1.54	16	19	83
IBM, San Jose	24	*20*	1.58	16	8	64
Oracle, Nashua	16	*11*	1.88	7	9	33
IBM, Toronto	16	*9*	1.94	11	5	26
IBM, San Jose	29	*21*	2.07	10	19	73
Microsoft, Redmond	24	*17*	2.00	5	19	58
Univ. of Toronto	20	*13*	1.90	8	12	25
AT&T	19	*8*	2.58	11	8	18
WPI, Worcester	17	*14*	1.35	9	8	42

Table 5: Institutions by research, industrial, demo

4.2 Authors

As authors drive the productivity of their affiliation, we finally investigate which authors contributed most research, industrial, and demo papers (Table 7). Some authors published their papers for up to four different affiliations. Regarding research publications, the two most prolific authors come from industry labs (Microsoft, AT&T). The other authors in the re-

Institution	pubs	*frac*	inst/p	1^{st}	2^{nd}	aut
ETH Zurich	29	*23*	1.52	11	18	39
Aalborg University	26	*14*	1.96	11	15	18
Univ. of Edinburgh	25	*15*	2.16	5	20	16
INRIA Le Chesnay	23	*12*	2.35	20	3	33
Univ. of Athens	22	*12*	2.32	6	16	17
MPI Saarbrücken	17	*12*	1.71	2	15	9
CWI Amsterdam	14	*12*	1.43	5	9	15
Univ. of Munich	12	*9*	1.67	10	2	20

Table 6: Top European research institutions

search top ten are mostly from American and Asian universities. Some leading affiliations (IBM San Jose, Stanford) have no author in the top ten for research papers. The authors with most demos are primarily from universities that apparently emphasize building of prototypes. There are significant differences in the average number of authors per paper. For the most prolific authors of research papers, this value is generally higher than for all research papers (average 3.43, Table 2) and the highest value is noted for the top listed researcher D. Srivastava. On the other hand, S. Chaudhuri has a relatively low (and below average) number of co-authors indicating that a very high productivity can also be achieved with a moderate level of cooperation.

5. CONCLUSIONS

We analyzed the author affiliations of database publications that appeared in four top venues in the last decade. We observed that most papers originate from US institutions and that industry labs run by IBM, Microsoft, and AT&T are most prolific. Asian institutions have achieved a comparable research output as European institutions. Top universities from Singapore and Hong Kong have reached a similar number of publications as the traditionally strong US universities Stanford, Wisconsin, and Berkeley.

Almost all research papers are co-authored; half of them involve at least two affiliations and almost a quarter two or more countries. The most frequent cross-national co-authorships occur between USA and Canada; Singapore has frequent cooperations with USA and China. The most prolific affiliations have relatively large teams; a high degree of collaboration also tends to improve the publication counts in the considered top venues. Europe hosts many but mostly small affiliations that do not yet achieve the paper counts of the top affiliations world-wide.

In future work, we plan to evaluate additional aspects such as affiliation-specific citation counts.

Author *affiliations*	pubs	*frac*	aut/p	1^{st}	2^{nd}
D Srivastava *AT&T*	39	*10*	4.31	20	19
S Chaudhuri *Microsoft Redmond*	38	*13*	3.00	19	19
N Koudas *AT&T, UW Seattle, Univ. Toronto*	36	*10*	3.81	11	25
MN Garofalakis *Bell, UC Berkeley, Yahoo, TU Crete*	35	*12*	3.29	19	16
Y Tao *HKUST, CityU of HK, CMU Pittsburgh, CUHK, Univ. of HK*	34	*12*	3.38	13	21
J Han *SFU Vancouver, U of I at Urbana-Champaign*	34	*10*	3.74	13	21
HV Jagadish *UMich Ann Arbor, AT&T*	33	*11*	3.61	15	18
D Papadias *HKUST*	32	*10*	3.41	16	16
R Ramakrishnan *UW-Madison, Yahoo*	32	*9*	4.09	6	26
BC Ooi *Natl. Univ. of Singapore*	32	*8*	3.97	11	21
MJ Carey *Propel Software San Jose, BEA San Jose, UC Irvine*	9	*3*	6.67	4	5
C Galindo-Legaria *Microsoft Redmond*	8	*3*	3.50	5	3
M Poess *Oracle Redwood Shores*	7	*3*	2.57	3	4
E Rundensteiner *WPI Worcester*	17	*4*	5.18	9	8
G Weikum *MPI Saarbr., Univ. Saarbrücken*	12	*3*	4.67	4	8
J Pei *SFU Vancouver, SUNY Buffalo*	11	*3*	4.73	7	4

Table 7: Authors by research, industrial, demo

6. REFERENCES

[1] Aumüller, D., Rahm, E.: Web-based Affiliation Matching. *Information Quality (ICIQ 09),* 2009

[2] Egghe, L., et al.: Methods for accrediting publications to authors or countries: Consequences for evaluation studies. *JASIST*, 2000

[3] Egghe, L., et al.: Collaboration and productivity: an investigation in Scientometrics and in a university repository. *Collnet, scientometrics and information management,* 2008

[4] Franceschet, M.: A comparison of bibliometric indicators for computer science scholars and journals on Web of Science and Google Scholar. *Scientometrics*, 2010

[5] Hagen, N.T.: Harmonic publication and citation counting: sharing authorship credit equitably – not equally, geometrically or arithmetically. *Scientometrics*, 84, 2010

[6] Nascimento, M., et al.: Analysis of SIGMOD's co-authorship graph. *Sigmod Record*, 2003

[7] Rahm, E.: Comparing the scientific impact of conference and journal publications in computer science. *Information Services and Use* 28 (2), 2008

[8] Rahm, E., Thor, A.: Citation analysis of database publications. *Sigmod Record*, 2005

[9] Sidiropoulos, A., et al.: Generalized Hirsch h-index for disclosing latent facts in citation networks. *Scientometrics* 2007

Report on Data-intensive Software Management and Mining

Seung-won Hwang

Department of Computer Science and Engineering
POSTECH, Korea
swhwang@postech.ac.kr

1. MOTIVATION

With the increase of publicly available software repositories, we face new challenges related to managing and mining large-scale software repositories. These challenges naturally call for novel research contributions from large-scale data management and mining researchers. Software repositories include multi-version source code, bug reports, and development history, and managing these repositories has been recognized as an important task in software design, development, and maintenance processes.

Due to these emerging challenges, many recent publications in major software engineering conferences and journals, such as ICSE, FSE, or TSE, have been contributed by database and mining researchers. To showcase some such efforts, mining approaches have been used for a corpus of failing traces, automatically collected from users experienced crashes. This corpus can assist developers to diagnose faults, prioritize handling of those faults, and locate faults in the code. Assuming similar faults have similar traces, mining approaches have been applied to mine similar past traces [5]. Once found, the fault location of the past trace can then be used to predict the location of the given trace. Similarly, developers, looking for related code developed in the past for reference, want an instant search of clones, which is essentially a ranked similarity search problem [4]. These efforts have been reported to increase the productivity of software developers, while enhancing the quality of code produced at the same time. Related efforts have also been reported in major human computer interaction conferences, such as SIGCHI. One example of recent research is Blueprint [2], which developed interfaces integrating instant source code and document searches for developers' code development sessions. Another example is Code Bubbles [1], proposing an interface to help developers effectively navigate code fragments spread all over a large corpus.

As these examples show, we observe emerging data management challenges from software problems that are highly relevant to database researchers.

- **O1: Software as a web-scale repository** A recent study [6] shows that an open-source code corpus can be as large as 420 million lines of source code and most code shares a similar structure, which positively suggests that mining for related code can be effective most of the time. However, this study reports that such analysis can take up to four months of CPU time, which invites effective indexing and mining techniques for large-scale repositories.

- **O2: Software as an evolving repository** However, source code repositories are reported to evolve over time. Such evolution has been noticed in software engineering research, and for evolving repositories, versioned index structures [3] studied in database research may enable efficient retrieval of evolutions.

- **O3: Software as a complex data structure** Many analysis schemes represent a piece of software code using different data structures. A parse tree is one example and a call graph is another. Alternatively, it could be high-dimensional vectors [4]. In all cases, an efficient similarity search is non-trivial and database techniques for complex data retrieval can be highly relevant.

- **O4: Software as a social network** Meanwhile, source code is reported to be a small part of a big underlying repository, where code is interconnected with related documents (bug reports or testing documents) and also people in charge of software segments. This forms a large-scale development social network [1].

[1] http://code.google.com/apis/opensocial/

The aim of the DSMM workshop, co-located with CIKM, was to draw the attention of database researchers to emerging challenges in large-scale software repositories. In particular, key topics discussed in this workshop included:

- Software indexing and search techniques.
- Software pattern mining and management techniques
- Management/mining of large-scale multi-version source code corpora
- Case studies of management/mining of large-scale software

2. WORKSHOP OUTLINE

This workshop was chaired by Audris Muckus (Avaya Labs, USA), Seung-won Hwang (POSTECH, Korea), and Sunghun Kim (HKUST, China). The PC team consisted of 15 reviewers from research labs and universities. Geographically, 10 reviewers were from North America, 3 from Europe and 2 from Asia. Half of the reviewers came from the software engineering community and the rest from the database community. The workshop consisted of three sessions, presented by one invited speaker and six paper presenters, as we discuss in detail below. A full list of speakers can be found at: http://ids.postech.ac.kr/dsmm.

2.1 Invited Session

The keynote speech, entitled "Developing accurate risk models requires mathematics, domain knowledge and common sense, although not necessarily in that order?" was delivered by Brendan Murphy from Microsoft Research, Cambridge. He addressed the challenge **O1** and presented on the difficulty of developing a risk model and getting a set of experienced engineers to accept and use your risk model. He then shared his experiences of developing such a model for the Microsoft Windows operating system and how getting engineers to accept the model is only possible by understanding the limitations of the model developed. His speech provided the participants with insights into how industry software is actually developed and how mathematics may not necessarily solve all problems encountered from industry development processes.

2.2 Research Session I: Software Development Process

This session had three papers on how data management techniques can assist software development processes.

The first presentation, entitled "Tree-pattern-based duplicate code detection", discussed the problem of automatically and accurately finding code clones in program files. In particular, this paper addressed the challenge **O3** and abstracted the problem as tree-pattern mining and clustering. However, comparing every pair can be costly and the authors present an efficient algorithm for this problem. Specifically, clustering enables flexible clone detection, covering four known types of clones: exact, parameter-substituted, structure-substituted, and non-contiguous clones.

The second presentation, entitled "Mining software repositories for bug detection requires accurate techniques of identifying bug-fix revisions", discussed repositories of bugs and their fixes. This talk was related to the challenge **O2** of managing evolving repositories of bug fixes. If a bug can be found the repository and annotated with how it was fixed in the past, we can automatically fix the bug guided by the existing fix. As a preliminary to this ambitious goal, the presenter discussed existing noises in the repository and how to remove them automatically. Specifically, the authors inspected bug-fix revisions of three open source projects (Eclipse, Lucene, and Columba) and categorized noises into 11 patterns. With these patterns identified, noise elimination can be automated.

The third presenter could not present her work, entitled "A case study on Model Driven Data Integration (MDDI) for Data Centric Software Development". According to her paper, her work described MDDI, which is a data integration problem over heterogeneous schemas, and presented challenges related to the problem, using actual historical data collected from universities. This paper was related to the challenge **O3**.

2.3 Research Session II: Large-scale Software Corpus

This session had three papers, related to maintaining and mining software-related corpora.

The first presentation, entitled "Transaction synchronization protocol using XML in client-Server environment", discussed versioned-repositories, such as software repositories that are constantly revised (and thus discussed the challenge **O2**). To keep these repositories consistent, transaction synchronization is an important problem: When clients modify their local database, writing this transaction back to the server and to all other clients requires transaction synchronization. To achieve reliable and efficient transaction synchronization, the presenter described an efficient protocol for trans-

actional synchronization, using XML as a transfer medium. This approach can significantly reduce the cost of data transfer between server and clients, thus increasing efficiency.

The second presentation, entitled "A method of workload compression based on characteristics for index selection", discussed a corpus of database workloads for development-related data that are frequently queried by developers. While managing such a workload provides important hints for optimizing the future queries, keeping all the workloads incurs prohibitive storage overheads. The presenter proposed an effective compression algorithm, which can significantly reduce the storage, without seriously compromising optimization quality for future queries. Such compression can be useful for a large-scale repository. This work discussed the challenge **O1**.

The last presentation, entitled "A targeted web crawling for building malicious javascript collection", raised the interesting question of whether the detection of malicious javascript code can be enhanced with a large corpus of malicious scripts. This problem is important, as malicious javascript frequently serves as a starting point for web-based attacks, in particular cross-site scripting. However, collecting up-to-date repositories of scripts is a non-trivial task. The presenter proposed a web crawler focused on more likely locations of malicious scripts for this task, and showed how this targeted web crawler performs. This talk was related to the challenge of large-scale repositories **O1**, but tackles the problem of "collecting" a repository with focus, namely malicious Javascripts.

3. CONCLUSIONS

DSMM was the first CIKM-associated workshop addressing the challenges of large-scale software repositories. Among the experts gathered in three different areas– program analysis, software engineering, and databases– the following future directions were discussed.

First, these challenges call for interdisciplinary efforts. To encourage such efforts, workshops of an interdisciplinary nature are useful. Though such interdisciplinary workshops have been founded, they are usually co-located at conferences related to one specific domain and attendance tends to biased toward that domain. It is thus still challenging to invite multi-disciplinary papers and presentations to one venue. Other efforts that do not require participants to gather in one location can be explored.

Second, for research of this nature to accumulate, releasing standard repositories, tasks, or benchmark data should be encouraged and rewarded. Having these common data, if successful, may address the first challenge as well– different teams can submit their work on common tasks and publish in journals. The participants expressed shared interest in forums for continuing interdisciplinary discussions on software repository management.

4. REFERENCES

[1] Andrew Bragdon et. al. Code bubbles: a working set-based interface for code understanding and maintenance. In *SIGCHI*, 2010.

[2] Joel Brandt et. al. Example-centric programming: Integrating web search into the development environment. In *SIGCHI*, 2010.

[3] Mark Gabel and Zhendong Su. How unique is source code. In *VLDB*, 2010.

[4] Mu-woong Lee, Jongwon Roh, Seung-won Hwang, and Sunghun Kim. Instant code clone search. In *FSE*, 2010.

[5] Chao Liu and Jiawei Han. Failure proximity: A fault localization-based approach. In *FSE*, 2006.

[6] Rui Zhang and Martin Stradling. The hvtree: a memory hierarchy aware version index. In *FSE*, 2010.

Dagstuhl Seminar on Bidirectional Transformations (BX)

Zhenjiang Hu
National Institute of Informatics, Japan
hu@nii.ac.jp

Andy Schürr
Technische Universitat Darmstadt
andy.schuerr@es.tu-darmstadt.de

Perdita Stevens
University of Edinburgh
perdita@inf.ed.ac.uk

James F. Terwilliger
Microsoft Corporation
james.terwilliger@microsoft.com

1. OVERVIEW

The Dagstuhl BX seminar, held January 16–21, 2011, brought together researchers from 13 countries across disciplines that study bidirectional transformations. It was a follow-up of the GRACE International Meeting on Bidirectional Transformations held in December 2008 near Tokyo, Japan [5]. This consisted of short introductions from each of the participants on their background and work, followed by presentations and demonstrations on representative technologies from each field, and some open discussion time. A major benefit of the GRACE meeting was the opportunity for the disciplines to get some initial exposure to each other.

The Dagstuhl seminar intended to go a step further and begin to identify commonalities between the disciplines and start to set a cross-disciplinary research agenda. The first part of the seminar consisted of tutorials from each of the four represented disciplines. The second part consisted of cross-disciplinary working groups dedicated to investigating specific examples of commonality between solutions or identifying requirements, terminology, or scenarios that may reach across fields. There were also sessions in which participants gave position statements on their own work.

Participants at both the Dagstuhl and GRACE seminars came from four disciplines: (1) Programming Languages, (2) Graph Transformations, (3) Software Engineering, and (4) Databases. At Dagstuhl, each of the first three disciplines made up about 2/7 of the participants, while databases took the remaining 1/7 out of about 45 participants. Representation from the database field was, nevertheless, an improvement over the turnout from the GRACE meeting.

2. TUTORIALS

The seminar opened with in-depth tutorials by representatives from each of the four disciplines on the various solutions to bidirectional transformation problems offered by that discipline. In this section, we present an overview of the material presented at those sessions.[1]

Programming Languages (PL)

Recently, many linguistic approaches have been proposed for describing bidirectional transformations in the programming languages community [6, 3]. An easy but non-ideal approach is to write a pair of transformation functions, one in each direction. Although this approach can use existing languages and works for simple transformations, scalability and maintenance are difficult. Other approaches specify both transformations with a single description. This tutorial surveyed recent work and presented promising approaches in detail.

First, Nate Foster gave an overview of design choices in languages for bidirectional transformations and introduced a common vocabulary for comparing different approaches. Important design choices include state-based versus operation-based updates, the set of constraints on the forward and backward transformations to ensure that they work well together, whether to allow freedom to choose a backward transformation for a forward transformation, and whether the backward transformation should handle every pair of data sources.

Second, Robert Glück described *reversible computing and reversible languages* [18]. Reversible computing is the study of computing models that exhibit both forward and backward determinism, while reversible languages are used to describe injective functions that can be effectively inverted. Reversible languages include explicit postcondition assertions,

[1]For a complete list of all presentations and working groups held at the seminar, consult the Dagstuhl site at http://bit.ly/dagstuhl-bx.

the ability to "un-call" a reversible procedure, and the possibility of clean and garbage-free computation of injective functions.

Janis Voigtländer described a *complement-based approach* to bidirectionalization [17] that automatically constructs a backward function from a forward function based on derivation of a constant-complement. Three methods are introduced: algorithmic generation based on the syntactic representation of the forward function, a semantic approach from a polymorphic forward function using parametricity and free theorems, and an integration of the previous two.

Next, Benjamin Pierce described *lens combinators* [3], the use of type systems to establish well-behavedness, and the issues surrounding the handling of ordered data. The fundamental concepts of bidirectional programming were explored in the simplest imaginable setting, where data are strings, types are regular expressions, and computation is finite state transduction. Their design emphasizes both robustness and ease of use, guaranteeing totality as well as strong well-behavedness conditions formulated as round-tripping laws.

Finally, Zhenjiang Hu introduced *trace-based bidirectionalization*, which is used for *bidirectionalizing graph transformations* [10]. This talk explained the basic idea of trace-based bidirectionalization, and showed how to compute traces effectively and how to propagate the view updates to the original database safely.

Databases (DB)

The *query-defined view* is a widely studied way to express a relationship between two models. Using queries to define model relationships has several benefits, including well-understood formal properties of query languages, the availability of robust and mature implementations, and the convenience of using the same language to specify both queries and mappings.

Despite copious research (e.g., [2]), practical support for updatable views in commercial systems falls well short of what research offers. Usability is among the reasons for this. Many developers prefer to specify the reverse mapping to a view manually as triggers. Also, it is difficult to provide constructive feedback as to why a given query is not reversible and how to fix the problem.

The database tutorial, presented by Jean-Luc Hainaut, Anthony Clève, and James Terwilliger, focused on alternatives for constructing updatable or bidirectional mappings. For instance, an *Object-Relational Mapping* (ORM) constructs an updatable object-oriented view of relational data. ORM tools are limited to transformations that are updatable (e.g., horizontal and vertical partitioning) and useful for bridging the object-relational impedance mismatch (e.g., inheritance mapping strategies like Table-per-Hierarchy) [12].

Data Exchange mappings are expressed using queries in a subset of first-order predicate calculus [1]. Research on inverting such mappings has expanded to include definitions of invertibility beyond those covered by classical updatable views. A mapping $\mathcal{M}$ may have an inverse $\mathcal{M}^{-1}$ where $\mathcal{M}^{-1} \circ \mathcal{M}$ is not the identity, but $\mathcal{M} \circ \mathcal{M}^{-1} \circ \mathcal{M} \equiv \mathcal{M}$. At least three such notions of inverse have now been studied, and each has relative advantages and disadvantages.

One characteristic that query-defined views, ORMs, and data exchange mappings all have in common is a top-down, holistic specification. The alternative approach to specifying a bidirectional mapping declaratively is to construct a mapping out of incremental components. Each component has proven bidirectional properties (not unlike a lens [4]), as well as other properties that make the component well-suited to the particular domain. For instance, in the *channels* framework, each component is capable of transforming schema evolution primitives and constraint declarations as well as queries and updates against its view schema [16].

A prime example of a component-based solution is *DB-MAIN*, an approach for managing multiple model artifacts even across levels of abstraction [8]. It can establish a relationship between a conceptual model, its corresponding logical model, and that model's corresponding physical model. Components in the *DB-MAIN* model describe how to translate constructs in one level of abstraction into another, and can thus handle most of the operations typically handled by ORMs.

Graph Transformations (GT)

Graph grammars were invented in the 1970s as a generalization of Chomsky Grammars. Graph transformation tools are often used for the formal specification of in-place model transformations or unidirectional transformations between different modeling and graph languages. *Triple Graph Grammars* (TGGs) have been developed for declarative and rule-based description of bidirectional transformations between related graph languages [15]. Formally, a TGG describes a language of graph triples with the first components being elements from the source language, the second components being instances of traceability relationships between source and target language elements, and the third com-

ponents being elements from the target language.

The first part of the GT tutorial, given by Andy Schürr, introduced the basic ideas of, and motivations for, the TGG formalism. It allows for the high-level description of functional and non-functional relationships between pairs of graphs. A family of graph transformations is derived from such a TGG specification that supports batch transformation and incremental change propagation scenarios in both directions as well as checking the consistency of given pairs of graphs. Traceability relationships between elements of related pairs of graphs are created and updated as a side effect [11]. The first part of the tutorial thereby prepared the ground for the second part presented by Frank Herrmann. This part sketched the formal background of TGGs based on category theory and related techniques for the verification of important properties of TGGs and derived graph translators. It also introduced a number of analysis techniques for the formal verification of desirable properties of TGGs including:

- *Correctness* which guarantees that graph tuples resulting from generated graph translators are instances of the relevant TGG language
- *Completeness* which guarantees that the generated graph translators can always translate updates of schema-compliant source graphs into updates of schema-compliant target graphs (and vice-versa)

Related techniques are also used for detection and resolution of conflicts between different translation rules and for improving the efficiency of generated engineering tool integrators [9].

Software Engineering (SE)

Model transformations are a key ingredient of model-driven development[13], a paradigm in which most design decisions are embodied in (graphical) models rather than in (textual) code. Several different models may be used in conjunction, each in a notation chosen to suit a particular task. Models, and code, may be partly or completely generated from other models using transformations. In the simplest case, a transformation may be just a kind of compilation: for example, code may be generated from a model by a transformation, and if the model changes, the code can be generated afresh.

As Krzysztof Czarnecki explained, the question of whether an individual model is valid is already quite complex; a model must meet both local structural requirements and constraints which may be global. One of the challenges for a model transformation language, even in the relatively simple unidirectional case, is that the transformation must not modify a model in such a way as to make it cease to conform to its metamodel.

The central challenge for this Dagstuhl meeting, however, was bidirectionality. There will typically be human input into both models, and each model will embody information that is not representable in the other. That is, this typical situation is *symmetric*. The job of the transformation is now to *maintain consistency*, bidirectionally. An important subproblem is to be able to check consistency – bearing in mind that the consistency relation may be non-bijective – and for this reason bijective model transformation formalisms tend to be relational in the sense that the consistency relation is more clearly apparent from the notation than the procedures for restoring consistency.

Relations in such a formalism (e.g., QVT-R [14]) are superficially similar to rules in graph transformations. There is a notion of matching a pattern in a model, that is, identifying a part of a model which is relevant to a particular relation; each match provokes some check of, or modification to, the other model. Beyond this opinions and formalisms differ widely. There may or may not be a well-defined corresponding part of the other model, which may or may not be recorded in some kind of linking structure. A linking structure may be formally defined as part of the definition of the transformation language, may be left implicit, or may have to be constructed by a human with heuristic tool support.

A related issue is the use, or not, of a record of the changes made to a model. Such a record can ease consistency maintenance at some pragmatic cost. As one point in the space of design possibilities, Stephan Hildebrandt introduced the language MoTE [7], which uses a combination of explicit link information and change notification to manage consistency.

3. WORKING GROUPS

What follows is a sampling of the seven working groups that met, and some key conclusions from them.

Taxonomy and Scenarios

Two groups focused on documenting commonality across disciplines by examining common terminology and scenarios. Despite working towards similar and sometimes identical research goals, the words used to describe concepts in those disciplines were vastly different. For instance, the word “model” in one discipline corresponds to “meta-model” in another discipline, and to “instance” in yet another discipline. Also, for a given scenario — say, object-

relational mappings — different disciplines had different pivot points of research; GT focuses on managing model relationships via grammar rules expressed at the meta-model level, while DB focuses on managing instances via formal properties expressed at the model level. The documentation of these groups will hopefully serve as fodder in future workshops on establishing a bidirectional transformation benchmark.

Mathematical Foundations of Lenses

The group on mathematical foundations of lenses served as a tutorial to most of the group on the connections between lenses and mathematical formalism. Michael Johnson demonstrated how lenses can be represented mathematically either as monads or as co-monads. Using these mathematical formalisms, one can prove why (and when) certain properties of lenses are important. One can also leverage monads to prove or discover properties of lenses that might not be immediately apparent.

One unfinished line of thought surrounded a potential link between lenses and data exchange. Database literature formalizes a data exchange mapping using predicate calculus, but practitioners outside the database field sometimes formalize similar mappings as a pair of adjacent lenses with opposing polarity. A mathematical link between lenses and data exchange may be intrinsically interesting, and might yield new results when considered within monadic formalism as well.

Reversible Programming and Graph Transformations

This group started with a broader discussion and classification of concepts for specifying and implementing a bidirectional programming language. Janus was, among other things, used as a running example for this purpose. The focus later turned towards a comparison of the pros and cons of BX programming languages like Janus on one hand and TGGs on the other hand. The conclusion was drawn that languages like Janus are well-suited for handling projections and arithmetic operation, but have problems handling complex data structures. TGGs are exactly the opposite, so the combination of these two lines of research seems promising.

4. FUTURE WORK

We went to Dagstuhl knowing that longer-term ideas like a common research agenda or benchmark would take more than a single week. The participants decided on several follow-up actions to keep work progressing:

- A follow-up meeting in the same style as GRACE and Dagstuhl to continue collaborating on a cross-disciplinary research agenda
- Workshops at conferences associated with each discipline to work toward specific, targeted goals (a first one has already been scheduled associated with GTTSE 2011, and will focus on developing a benchmark[2]; a second follow-up event has just been accepted as a satellite workshop for ETAPS)
- Tutorials and other education-minded events at conferences to continue bringing awareness of bidirectional solutions from other disciplines, as well as awareness of the general BX effort
- Smaller-scale research cooperations that combine techniques from different fields like merging concepts from bidirectional programming languages and triple graph grammars as envisaged in one of the seminar's working groups.

In particular, a goal of the upcoming seminars and workshops is to increase database community participation. The bidirectional transformation problem has origins deep in the database community [2], but has grown so that solutions are being driven from many directions in different fields across computer science. The plan is to hold some of the tutorials or workshops at database venues to help solicit ideas and opportunities for collaboration; details will be made available once they are scheduled.

5. REFERENCES

[1] M. Arenas, P. Barceló, L. Libkin, F. Murlak. Relational and XML Data Exchange. *Synthesis Lectures on Data Management*.

[2] F. Bancilhon, N. Spyratos. Update Semantics of Relational Views. *ACM Transactions on Database Systems*, December 1981, 6(4).

[3] A. Bohannon, J. N. Foster, B. C. Pierce, A. Pilkiewicz, A. Schmitt. Boomerang: resourceful lenses for string data. *POPL 2008*.

[4] A. Bohannon, B. C. Pierce, J. A. Vaughan. Relational lenses: a language for updatable views. *PODS 2006*.

[5] K. Czarnecki, J. N. Foster, Z. Hu, R. Lämmel, Andy Schürr, J. F. Terwilliger. Bidirectional Transformations: A Cross-Discipline Perspective. *ICMT 2009*.

[6] J. N. Foster, M. B. Greenwald, J. T. Moore, B. C. Pierce, A. Schmitt. Combinators for bi-directional tree transformations: a linguistic approach to the view update problem. *POPL 2005*.

[2]http://www.di.univaq.it/CSXW2011/

[7] H. Giese, S. Neumann, S. Hildebrandt. Model Synchronization at Work: Keeping SysML and AUTOSAR Models Consistent. *Graph Transformations and Model Driven Enginering — Essays Dedicated to Manfred Nagl on the Occasion of his 65th Birthday*, LNCS 5765.

[8] J.-L. Hainaut. The Transformational Approach to Database Engineering. *GTTSE 2006*.

[9] F. Hermann, H. Ehrig, F. Orejas, U. Golas. Formal Analysis of Functional Behaviour of Model Transformations Based on Triple Graph Grammars. *Proc. Int. Conf. in Graph Transformation ICGT 2010*.

[10] S. Hidaka, Z. Hu, K. Inaba, H. Kato, K. Matsuda, K. Nakano. Bidirectionalizing graph transformation. *ICFP 2010*.

[11] F. Klar, M. Lauder, A. Königs, A. Schürr. Extended Triple Graph Grammars with Efficient and Compatible Graph Translators. *Graph Transformations and Model Driven Enginering — Essays Dedicated to Manfred Nagl on the Occasion of his 65th Birthday*, LNCS 5765.

[12] S. Melnik, A. Adya, P. A. Bernstein. Compiling mappings to bridge applications and databases. *ACM Trans. Database Syst.*, 33(4).

[13] Object Management Group. MDA Guide V1.0.1. *omg/03-06-01*

[14] Object Management Group. Queries, Views and Transformations. *formal/2011-01-01*

[15] Specification of Graph Translators with Triple Graph Grammars. *Proc. Int. Workshop on Graph-Theoretic Concepts in Computer Science (WG 1994)*.

[16] J. F. Terwilliger, L. M. L. Delcambre, D. Maier, J. Steinhauer, S. Britell. Updatable and Evolvable Transforms for Virtual Databases. *PVLDB 3(1)*.

[17] J. Voigtländer, Z. Hu, K. Matsuda, M. Wang. Combining syntactic and semantic bidirectionalization. *ICFP 2010*.

[18] T. Yokoyama, H.B. Axelsen, R. Glück. Reversible flowchart languages and the structured reversible program theorem. *ICALP 2008*.

Report on DEIS'10: Advanced School on Data Exchange, Information, and Streams (A GI-Dagstuhl Seminar)

Phokion G. Kolaitis* Maurizio Lenzerini† Nicole Schweikardt‡

1. INTRODUCTION

Most computer scientists are familiar with the Schloss Dagstuhl - Leibniz Center for Informatics or, simply, Dagstuhl as the place "where computer scientists meet". Over the years, literally thousands of computer scientists have attended one or more Dagstuhl seminars in which participants spend a week interacting with colleagues in an informal setting by sharing new results and work in progress, exchanging ideas, or embarking on new collaborations. Alongside these year-round seminars, however, Dagstuhl also hosts a different kind of event that is expressly geared towards students and postdoctoral scholars. Specifically, Dagstuhl is also the home of the GI-Dagstuhl Seminars[1], which are sponsored jointly by the German Society for Informatics (GI) and the Schloss Dagstuhl - Leibniz Center for Informatics. The designated purpose of GI-Dagstuhl Seminars is to enable young researchers to learn about new developments in a particular area of research through active engagement in the seminar, which is typically organized by an international team of senior researchers. GI-Dagstuhl Seminars take place by far less frequently than regular Dagstuhl-Seminars; actually, only one or two such seminars has taken place each year during the past six years. Furthermore, GI-Dagstuhl Seminars are limited to at most 20-25 participants, including the organizers.

This paper reports on GI-Dagstuhl Seminar 10452, an Advanced School on Data Exchange, Integration, and Streams (DEIS'10), which took place from November 7 to November 12, 2010 and was organized by the three authors.

2. SCIENTIFIC THEME

DEIS'10 focused on data exchange, data integration, and data streams. These are three different, yet inter-related, facets of information integration that have been investigated in depth by the research community in recent years.

Both data exchange and data integration deal with the execution of information integration, but they adopt distinctly different approaches. Data exchange is the problem of transforming data residing in different sources into data structured under a target schema; in particular, data exchange entails the materialization of data, after the data have been extracted from the sources and re-structured into a unified format. In contrast, data integration can be described as symbolic or virtual integration: users are provided with the capability to pose queries and obtain answers via the unified format interface, while the data remain in the sources and no materialization of the restructured data is required.

In the basic data stream model, the input data consists of one or several streams of data items that can be read only sequentially, one after the other. This scenario is relevant for a large number of applications where massive amounts of data need to be processed. Typically, algorithms have to work with one or few passes over the data and a memory buffer of size significantly smaller than the input size.

3. PROCESS AND TIMETABLE

In response to a call for proposals for GI-Dagstuhl Seminars, we submitted a proposal for DEIS'10 in November of 2009. In this proposal, we described the scientific theme of DEIS'10, listed the specialized topics, and also spelled out the procedure for selecting participants in this event. After our proposal was accepted, we disseminated the plan for DEIS'10 via postings to a number of forums, including DBWorld, and through a dedicated web page at `http://www.tks.cs.uni-frankfurt.de/events/deis10`

Potential applicants were asked to submit by July 15, 2010 an application consisting of a letter of in-

*UC Santa Cruz & IBM Research - Almaden, USA
†Università di Roma La Sapienza , Italy
‡Goethe-Universität Frankfurt am Main, Germany
[1]`http://www.dagstuhl.de/en/program/gi-dagstuhl-seminars/`

terest, a curriculum vitae, up to three representative papers or theses authored by the applicant, and a letter of recommendation from an academic supervisor or other senior colleague. We received 31 applications, out of which 22 applicants were selected to participate in DEIS'10; together with the organizers, this brought the total number of DEIS'10 participants to 25, which is the maximum that can be accommodated in a GI-Dagstuhl Seminar. The great majority of the applications received were of very high quality. In fact, we would have gladly accepted more applicants had there been more room. Of the 22 successful applicants, 18 were graduate students and 4 were postdoctoral scholars. In terms of geography, 18 were located in Europe, 3 in North America, and 1 in South America. We note that the participants were selected not on the basis of a paper they submitted, but, instead, on their ability and potential to participate in the event in a meaningful way.

The participants were notified of their selection on September 1, 2010. Each participant was asked to study the relevant literature in a specialized topic that was assigned to him or her by the organizers of DEIS'10, based on the interests and expertise of the participants. Each participant was assigned one of the three organizers as mentor. Mentors and mentees interacted via email during September and October 2010. In particular, participants were asked to send their mentors a progress report with an outline of their presentation on October 1, 2010, followed by a semi-final draft of the slides of their presentation on November 1, 2010.

During the first day of DEIS'10, each of the three organizers gave a 90-minute tutorial on one of the three main themes of the school. Specifically, there was a tutorial on "Schema Mappings and Data Exchange" by Phokion Kolaitis, a tutorial on "Data Integration" by Maurizio Lenzerini, and a tutorial on "Data Streams" by Nicole Schweikardt. The rest of the program consisted of the presentations by the participants. Each participant was given 45 minutes to present an overview of the specialized topic assigned to her or him; the presentations were followed by or were interspersed with questions by the audience, so that a total of one hour was allotted to each specialized topic.

While a small number of participants presented some of their own research work, most of the presentations were a synthesis of papers studied by the participants in the months before DEIS'10 took place. In total, well over 100 published papers were distilled and synthesized by the participants in their presentations. The slides of these presentations and the relevant bibliographical references can be found at the web page of DEIS'10, which was given earlier in this section.

In addition to the tutorials and the presentations of specialized topics, an after-dinner problem session was held in the second day of DEIS'10. In this session, both the organizers and the participants presented selected open problems in each of the three main themes of DEIS'10. The last time slot of DEIS'10 was a wrap-up session during which feedback about the event was solicited and tentative plans for a follow-up event were discussed.

4. ASSESSMENT

From the viewpoint of the organizers, DEIS'10 was both an unqualified success and a truly satisfying experience. In the past, each of us had participated in typical advanced schools in which instructors give week-long short courses on a topic of their expertise. In that setting, attendees choose which courses to attend and then, for the most part, passively absorb the technical material presented by the instructors. The interaction between the instructors and the audience is usually limited to the questions that are asked from time to time during the course. As a result, it is unlikely that instructors get to know the attendees (or, at least, the majority of the attendees) and to have a quality interaction with them. Our experience with DEIS'10 was very different. The attendees of DEIS'10 were active and engaged participants who worked hard to first study a specialized topic and then present an overview of the topic assigned to them at the school. Furthermore, we got to know the participants well by evaluating their applications, communicating with them on a one-to-one basis via email before the event, and then meeting them in person at Dagstuhl and in a setting that fosters interaction and open communication.

From the viewpoint of the participants, DEIS'10 seems to have been a very positive experience for them. We base this assessment on both the feedback we received during the wrap-up session at the end of DEIS'10 and on the summary of the written survey that the participants completed by filling the form that is distributed to all participants of Dagstuhl seminars. In addition to high numerical ratings to questions concerning the scientific quality of the event, the inspiration of new ideas, and the identification of new research directions, participants expressed appreciation for the "interactive, informal atmosphere" and the "high quality of talks". In fact, one participant went as far as declaring that the worst aspect of this seminar is

that "it's over".

As regards criticisms and suggestions for improvement, there was a consensus that "the schedule was too dense". As a remedy, some participants suggested having a mix of short and long presentations, instead of allocating the same amount of time to each participant. Others suggested cutting down the time of presentations uniformly for all participants. Of course, a different way to address this issue is to accept a smaller number of participants; however, this would be at the expense of turning down perfectly qualified and highly motivated applicants. Participants also suggested distributing open problems before the school and holding additional sessions during the school in which participants discuss open problems or present work in progress. Finally, at the scientific level, it was felt that some of the topics in the data streams area were disconnected from the topics in the other two areas of the school. Clearly, much remains to be done to strengthen the ties between research in data streams on the one side and research in data exchange and data integration on the other. Bringing people from all three areas together is the first step.

5. FOLLOW-UP

For some of the topics presented at DEIS'10, excellent survey articles already exist. Some other topics are still too nascent to justify survey articles. For several more mature topics for which no survey articles presently exist, we felt that the time is ripe to produce such survey articles. To this effect, we submitted a proposal to the Scientific Directorate of Dagstuhl to compile a volume consisting of state-of-the-art surveys of selected topics in data exchange, integration, and streams. Our proposal has now been accepted, and the planned volume will be published in the series *Dagstuhl Follow-Ups*, which is is a new open-access publication venue for peer-reviewed articles based on Dagstuhl seminars (see `http://www.dagstuhl.de/dfu`). The survey articles in this volume will be authored by a number of DEIS'10 participants, who have already been invited and agreed to work on this project.

6. ACKNOWLEDGMENTS

We are grateful to the Schloss Dagstuhl - Leibniz Center for Informatics and to the German Society for Informatics (GI) for giving us the opportunity to organize this event. In particular, we thank Dr. Marc Herbstritt, Member of the Scientific Staff of the Schloss Dagstuhl - Leibniz Center for Informatics, for his help and and encouragement at every step of this project. We also wish to acknowledge the German Research Foundation (DFG) and the ACM Special Interest Group on Management Of Data (SIGMOD) for their generous financial support, which made it possible to provide travel support to participants of DEIS'10. Finally, we wish to thank all participants for their hard work, engagement, and enthusiasm that made DEIS'10 a successful event.