SIGMOD Record – December 2005 – Volume 34 Issue 4 –

[**Editor's note**: With the exception of the last pages –which would be the back cover of the printed issue– that are not included in this file, it has the same contents as the printed edition. All the articles are also available individually online and have been put together here for convenience only.]

SIGMOD Officers, Committees, and Awardees

Chair
Raghu Ramakrishnan
Department of Computer Sciences
University of Wisconsin-Madison
1210 West Dayton Street
Madison, WI 53706-1685
USA
raghu@cs.wisc.edu

Vice-Chair
Yannis Ioannidis
University Of Athens
Department of Informatics & Telecom
Panepistimioupolis, Informatics Bldngs
157 84 Ilissia, Athens
HELLAS
yannis@di.uoa.gr

Secretary/Treasurer
Mary Fernández
ATT Labs - Research
180 Park Ave., Bldg 103, E277
Florham Park, NJ 07932-0971
USA
mff@research.att.com

Information Director: Alexandros Labrinidis, University of Pittsburgh, labrinid@cs.pitt.edu.

Associate Information Directors: Manfred Jeusfeld, Dongwon Lee, Michael Ley, Frank Neven, Altigran Soares da Silva, Jun Yang.

Advisory Board: Tamer Ozsu (Chair), University of Waterloo, tozsu@cs.uwaterloo.ca, Rakesh Agrawal, Phil Bernstein, Peter Buneman, David DeWitt, Hector Garcia-Molina, Jim Gray, Masaru Kitsuregawa, Jiawei Han, Alberto Laender, Krithi Ramamritham, Hans Schek, Rick Snodgrass, and Gerhard Weikum.

SIGMOD Conference Coordinator: Jianwen Su, UC Santa Barbara, su@cs.ucsb.edu

SIGMOD Workshops Coordinator: Laurent Amsaleg, IRISA Lab, Laurent.Amsaleg@irisa.fr

Industrial Advisory Board: Daniel Barbará (Chair), George Mason Univ., dbarbara@isse.gmu.edu, José Blakeley, Paul G. Brown, Umeshwar Dayal, Mark Graves, Ashish Gupta, Hank Korth, Nelson M. Mattos, Marie-Anne Neimat, Douglas Voss.

SIGMOD Record Editorial Board: Mario A. Nascimento (Editor), University of Alberta, mn@cs.ualberta.ca, José Blakeley, Ugur Çetintemel, Brian Cooper, Andrew Eisenberg, Leonid Libkin, Alexandros Labrinidis, Jim Melton, Len Seligman, Jignesh Patel, Ken Ross, Marianne Winslett.

SIGMOD Anthology Editorial Board: Curtis Dyreson (Editor), Washington State University, cdyreson@eecs.wsu.edu, Nick Kline, Joseph Albert, Stefano Ceri, David Lomet.

SIGMOD DiSC Editorial Board: Shahram Ghandeharizadeh (Editor), USC, shahram@pollux.usc.edu, A. Ailamaki, W. Aref, V. Atluri, R. Barga, K. Boehm, K.S. Candan, Z. Chen, B. Cooper, J. Eder, V. Ganti, J. Goldstein, G. Golovchinsky, Z. Ives, H-A. Jacobsen, V. Kalogeraki, S.H. Kim, L.V.S. Lakshmanan, D. Lopresti, M. Mattoso, S. Mehrotra, R. Miller, B. Moon, V. Oria, G. Ozsoyoglu, J. Pei, A. Picariello, F. Sadri, J. Shanmugasundaram, J. Srivastava, K. Tanaka, W. Tavanapong, V. Tsotras, M. Zaki, R. Zimmermann.

SIGMOD Digital Review Editorial Board: H. V. Jagadish (Editor), Univ. of Michigan, jag@eecs.umich.edu, Alon Halevy, Michael Ley, Yannis Papakonstantinou, Nandit Soparkar.

Sister Society Liaisons: Stefano Ceri (VLDB Foundation and EDBT Endowment), Hongjun Lu (SIGKDD and CCFDBS), Yannis Ioannidis (IEEE TCDE), Serge Abiteboul (PODS and ICDT Council).

Latin American Liaison Commitee: Claudia M. Bauzer Medeiros (Chair), University of Campinas, cmbm@ic.unicamp.br Alfonso Aguirre, Leopoldo Bertossi, Alberto Laender, Sergio Lifschitz, Marta Mattoso, Gustavo Rossi.

Awards Committee: Moshe Y. Vardi (Chair), Rice University, vardi@cs.rice.edu. Rudolf Bayer, Masaru Kitsuregawa, Z. Meral Ozsoyoglu, Pat Selinger, Michael Stonebraker.

Award Recipients:

Innovation Award: Michael Stonebraker, Jim Gray, Philip Bernstein, David DeWitt, C. Mohan, David Maier, Serge Abiteboul, Hector Garcia-Molina, Rakesh Agrawal, Rudolf Bayer, Patricia Selinger, Don Chamberlin, Ronald Fagin.

Contributions Award: Maria Zemankova, Gio Wiederhold, Yahiko Kambayashi, Jeffrey Ullman, Avi Silberschatz, Won Kim, Raghu Ramakrishnan, Laura Haas, Michael Carey, Daniel Rosenkrantz, Richard Snodgrass, Michael Ley, Surajit Chaudhuri.

Editor's Notes

This issue is especially important to me as it marks the completion of my first year as the *Record*'s Editor. So far so good, but I thought it would be a good time to think about the following question: "what is the *Record*'s role within ACM SIGMOD?" After some thinking (and three versions of these Notes!), I thought that this can be translated into the following questions: should the *Record* become a "real" fully refereed journal? Should it be more of a venue for articles, maybe not as "researchy" (I know this is not a word, but English is a forgiving language), but of more interest to the SIGMOD community at large?

I think the latter should be it. Actually that is what I think it has mostly been, and the regular columns are a good example of that. Winslett's Distinguished DB Profiles and Ross' Influential Papers columns are examples of what one could not find elsewhere but on the *Record*, and I have received good feedback about those in general. Çetintemel's Research Centers column helps most of us to place our colleagues' work and environment within a wider framework besides their papers, and Cooper's Articles, Reports and Notes column allows one to have a better idea about other events, in a world where there are already too many events to follow. This being said, it does not mean that there is no room for more research-oriented articles. There is, but the key is on their coverage. Libkin's Database Principles column, for instance, is a good example. It always features a research-oriented article but typically of a broader coverage than a typical, say, *PODS* paper. I dare to say that those papers would not make into *PODS* or *ICDT* (due to their broader scope, not quality!) but they are perfectly fine for the *Record*. (Similar reasoning applies to all other columns, the one above are only used as examples to motivate my idea.) In fact, this very issue is an excellent example of articles that I consider far-reaching contributions, more on that later. On this front, I believe, and have heard from many others, that the *Record* is doing its job well.

Now, let me venture into a more delicate terrain, what I think the *Record* is not. It is not to be viewed as a full-fledged archival journal. Yes, it does publish original research articles, which are reviewed but not refereed, i.e., not as rigorous as in a typical journal process. And at this point a fair question that you may have is: what is the difference between reviewing and refereeing? For that I refer to ACM's "Policy on Pre-Publication Evaluation"[1] which I took the liberty to summarize below:

- *Refereed material is subjected to a detailed peer review … Refereeing is generally directed to scholarly material for purposes of ascertaining originality, correctness, novelty, importance, and clarity of exposition …*
- *Formally reviewed material is subjected to a structured evaluation and critique procedure following a defined process uniformly applied as with refereeing, only without requiring that the tests of scholarly originality, novelty and importance be applied in the previous sense … Evaluation for technical accuracy is still required, and the criterion of clarity of exposition may be interpreted as readability by a certain intended audience.*
- *Reviewed material is subjected to a more informal and not necessarily uniform process of volunteer review, with standards dependent upon the publication and the type of material …*

Furthermore, as per ACM's policy "Publications consisting only of reviewed or unreviewed material are considered 'informal'" (which to bears no relationship to unimportant!) There is no positive or negative spin on this, this is ACM's policy, and the *Record* follows it.

[1] `http://www.acm.org/pubs/prepub_eval.html` (Thanks to Rick Snodgrass for pointing me out to this documentation).

This brings me to a side note, that may be important for those less experienced and/or perhaps more "aggressive". Once I was asked by a Department Chair if the *Record* was a *typical* archival journal. Clearly, my view is that it is not, and that conflicted with that of a faculty member in his/her department, incidentally seeking promotion. Since I am the Editor, I believe my definition prevailed, and the faculty member's credibility suffered with the exaggerated claim of having one too many journal papers. A word of advice, if I may, be careful with your claims with respect your publications on the *Record*!

So, what is the conclusion I am coming to? The *Record* should strive to be broader in its coverage, without losing sight of correctness and novelty. Articles of the survey type, or presenting state-of-the-art and research directions (or non-directions for that matter), position and visionary (not delusional!) papers –these should form the core of the *Record*. Special issues can also be accommodated, and in this case they should be more research-oriented towards a common topic of general interest. This being said, I think that perhaps I should avoid having in the *Record* papers that could very well fit in conferences or workshops. We all know that there ought to be a venue for every technical paper one can write. On the one hand, this is not to say that I will not accept research articles, at least not yet, but I will be stricter about the "coverage" aspect. On the other hand, that is to say that good papers may be rejected, because in my opinion (or of the reviewers) there is a better dissemination venue than the *Record* and/or its contribution is too narrow, despite being correct and original.

The above is mainly what I have in mind, and I would most certainly like to hear your opinions on the ideas above. The *Record* has to be what you want it to be and these ideas have to be fully debated among the *Record*'s associate editors and within SIGMOD's Executive Committee as well before any action can be taken. Your views would be important input to that discussion, so feel free to email me at `record@sigmod.acm.org` (I would appreciate if you use "SIGMOD Record Feedback" or something similar in the subject header).

Let me now give you some statistics about the *Record* since I took over its editorship. All four issues of 2005 consumed 396 pages –not bad considering that I was told the *Record* has budget of 400 pages/year! As of the time of this writing, I have received 30 article submissions. Those do not include invited or solicited contributions, articles I considered of general interest to SIGMOD's community or the special section in Sept./2005. Of those, 9 have been accepted for publication, 12 have been rejected and the remainder is under review. Given that the average submission has 6 pages, one can clearly see that the bulk of published material is made of the regular columns or those articles I would classify as of general interest, and which are not counted within the submitted articles above. While at it, let me thank all those who have helped review papers for the *Record*! They have helped a lot with their own time for the good of the community, and have also been quite forgiving with my requests and "reminders."

How about this issue though? Besides the usual columns and some submitted research articles, it contains a few articles that illustrate well the spirit of what I mean above. The report by Amer-Yahia, which I placed under the Event Reports column, is about one of the panels at SIGMOD 2005. I welcome very much Amer-Yahia's initiative and would like to encourage other panel organizers, not only of the SIGMOD conference, but of other events as well, to write such reports. Still on the spirit of broader coverage this issue also includes a paper by Gray and colleagues on scientific data management. This article has been published earlier on Cyber Technology Watch (February 2005), and is re-published here (with permission of the original publishers) because the authors felt that the *Record* provides the broad reachability and the right audience for this paper. The papers by Stonebraker, Çetintemel and Zdonik, as well as the one by Franklin, Halevy and Maier are also very much in line with what I mean by a broader article that is not as focused towards one contribution (as the vast majority of journal and conference papers) but is just as important as it can help shape one's research program.

The other papers I want to mention are likely to spark some discussion among many of us (somehow the word "heated" comes to mind). The first is by Rahm and Thor where they present an analysis of the citations within our community in the last 10 years using some of the well-known venues we have. I was particularly proud to see that the top-three most cited articles in an archival publication[2] have appeared in the *Record*. It is also important to note that those papers are survey papers, which again sends me back to the view that the *Record* is indeed a good venue for such broader papers. Note that certainly there is still place for more authoritative venues, though some of those are typically much slower in their turnaround time (I recently learned from an author whose paper is still under the first round of reviews after over two years!) The second paper is by Sidiropoulos and Manolopoulos, where they also use citation analysis to rank publications. Although the authors contextualize their discussion about "prize awarding" I suggest you to read it beyond that narrow goal. I should also say that the opinions and facts stated in both papers are those of the authors and not necessarily endorsed by ACM SIGMOD. Nevertheless, in my opinion, they help illustrate very well my view of the *Record*'s role within our community: to provide news and articles of interest to the community as whole. A third related paper is the one by Bernstein and his colleagues reflecting the VLDB 2005 panel on "database publication practices". It provides some interesting statistics about the paper submission process to some of our best known archival publications.

It is also time for my first "errata": in June's issue the paper by Hull and Su ("Tools for Composite Web Services: A Short Overview") was missing some references that were mentioned in the text. That escaped both Libkin's and my notice, and we apologize for the inconvenience. At the time you read this it should have been replaced online.

Finally, I have three thank-you notes, in no particular order: (1) to the ACM staff who handles the *Record's* actual publication, in particular to Julie Goetz, (2) to Alex Labrinidis to helping me putting the *Record* available via SIGMOD Online as soon as it is ready, and (3) to Ken Ross; this issue also marks his last issue as the editor for the Influential Papers column (he is deservedly moving on to be associate editor on *ACM TODS*, and I hope to soon have someone taking over as associate editor of that column). Oh, there is a fourth thank-you. This may well be the longest "Editor Notes" the *Record* ever had, and if you managed to read these notes up to this point, you certainly deserve a big thank-you for your interest!

Mario Nascimento, Editor.
October 2005

[2] Rahm and Thor refer to "journal" publications, however since I want to distinguish the Record from a "typical" journal, I chose to use the term "archival publication" instead of journal.

It is a pleasure and a privilege to write this first letter to you, the SIGMOD membership, in my newly-elected role of Chair. Mary Fernández, Yannis Ioannidis and I have been busy learning about ACM, SIGMOD, and our new jobs. A particular thank you is due to Tamer Özsu, Joachim Hammer and Marianne Winslett, the outgoing officers who have been generous with their time and experience in answering our numerous questions and helping out with timely advice.

Fortunately, much of the normal operation of SIGMOD is in the capable hands of a group of dedicated individuals who provide continuity across elections, so the transition has been smooth. This group, together with Mary, Yannis, myself, and Tamer (in his capacity as outgoing Chair), comprises the SIGMOD Executive Committee, and includes: Curtis Dyreson (SIGMOD Anthology Editor), Shahram Ghandeharizadeh (SIGMOD DiSC Editor), Georg Gottlob (PODS Liaison), Alex Labrinidis (Information Director), Mario Nascimento (SIGMOD Record Editor), and Jianwen Su (Conference Coordinator). Ginger Ignatoff is our ACM SIG liaison.

One of the first actions of the new Executive Committee was to invite several distinguished members of the community to serve on the SIGMOD Advisory Board, and I'm happy to announce that we now have a wonderful resource renewed. We will benefit greatly from their insight, institutional memory, and breadth of experience. The new board is:

Rakesh Agrawal, Phil Bernstein, Peter Buneman, David DeWitt, Hector Garcia-Molina, Jim Gray, Masaru Kitsuregawa, Jiawei Han, Alberto Laender, Tamer Ozsu (Chair), Krithi Ramamritham, Hans Schek, Rick Snodgrass, and Gerhard Weikum

I am also happy to announce that Beng-Chin Ooi has agreed to serve as the program committee chair of SIGMOD 2007, which will be held in Beijing from June 11 to 14. Jianwen Su and the general chairs, Tok Wang Ling and Lizhou Zhou have begun working on the arrangements for the conference. Meantime, arrangements are proceeding apace for SIGMOD 2006 as well.

Mary, Yannis and I are enjoying working together as a team, and I've invited them to share their thoughts with you as well.

Sincerely,
Raghu Ramakrishnan

Raghu, Yannis and I are off to a great start as your new SIGMOD officers, thanks to a lot of help from your former officers, the current executive committee, and the ACM staff. Luckily, all these capable people keep the SIGMOD SIG running smoothly while we learn how to do our jobs. We are supporting the organizing committees of the SIGMOD 2006, 2007, and even 2008 (!) conferences in their planning activities, and we are especially excited about our 2007 venue in Beijing, China. Once we are underway, I plan to propose that the SIGMOD EC establish a scholarship fund, possibly underwritten by industrial sponsors, to finance the attendance of professors and students at the SIGMOD/PODS conferences who would otherwise not be able to attend. Increasing the diversity of people and institutions involved in SIGMOD will help maintain its vibrance and relevance in the future.

Mary Fernández

The first few months of serving as SIGMOD Vice-Chair have been truly exciting. Working with Raghu and Mary, and surrounded by experienced colleagues in the Executive Committee, ACM, and soon the new Advisory Board, I feel the environment is ideal for continuing and further expanding the wonderful work of the former officers for the benefit of all SIGMOD members. The initial learning curve has been quite steep, but as things fall into place, our program is getting under way. Some of the first priorities are to strengthen the impact database research has on the rest of computer science and to increase the number of SIGMOD members. To serve both directions simultaneously, I intend to explore the possibility of clustering around SIGMOD various sister societies that are dedicated to specialized forms of data management, by forming strategic membership alliances with them and co-locating or even merging of future conferences.

Yannis Ioannidis

In Memoriam Alberto Oscar Mendelzon

July 28, 1951 - June 16, 2005

Alberto Oscar Mendelzon passed away on June 16, 2005 after a two-year battle with cancer. This tribute to Alberto and his achievements is written in recognition of his great intellect and his generous friendship. Both have influenced and inspired many in the database research community.

Alberto was one of the pioneers who helped to lay the foundations of relational databases. His early work on database dependencies has been influential in both the theory and practice of data management. He has made fundamental contributions in the areas of graphical query languages, knowledge-base systems, and on-line analytic processing. His work has provided the foundation for languages used to query the structure of the web. More than all of this, he was a man admired for his humor, his modesty, and his devotion to his students, his family, and his friends.

Alberto Mendelzon, professor of computer science at the University of Toronto, was born in Buenos Aires, Argentina. His academic journey began in Argentina and he maintained, throughout his life, close ties to his home country and home continent. He graduated from the University of Buenos Aires in 1973 before studying at Princeton as a Fulbright Scholar. At Princeton, he received a M.S.E. degree in 1977, a M.A. degree in 1978, and a Ph.D. degree in 1979. He was a post-doctoral fellow at IBM's T.J. Watson Research Center for a year before joining the University of Toronto in 1980.

Alberto established some of the earliest results on using the relational data model. Together with his thesis advisor, Jeff Ullman, and fellow Princeton students, including David Maier and Yeshoshua Sagiv, Alberto co-authored a number of influential papers that laid out the fundamental issues and approaches for relational databases. Alberto's 1979 Princeton Ph.D. on "Data Dependencies in the Relational Model" provided a foundation for understanding and reasoning about the consistency of data. The famous MMS79 paper (Maier, Mendelzon and Sagiv, TODS 1979), which was selected as one of the best papers of SIGMOD 1979 and invited for TODS publication, introduced the chase, a method for testing implication of data dependencies. This work has been highly influential: it is used, directly or indirectly, on an everyday basis by people who design databases, and it is used in commercial systems to reason about the consistency and correctness of a data design. New applications of the chase in meta-data management and data exchange are still being discovered.

In the 1980's, Alberto began an important line of work on graphical query languages. His work has been called *prescient* as it began before the World Wide Web, and nonetheless established many of the scientific principles required for designing languages to query the Web. His work in this area led to a series of well-known projects (Hy+, WebSQL, WebOQL, ToX). His work on WebSQL has been hailed as ground breaking. The highly cited MMM97 paper (Mendelzon, Mihaila and Milo, International Journal on Digital Libraries 1997) has inspired many follow-on papers on Web query languages.

Throughout his career, Alberto studied the deductive properties of data and knowledge bases. Particularly noteworthy are his contributions to understanding the semantics of updates in knowledge bases and the application of this to the problem of database updates. In work with Katsuno, Alberto resolved the important, and deep, issue of updating knowledge bases. By making a distinction between two kinds of modification, update and revision, they laid the foundation for studying updates within knowledge bases and established why updates are fundamentally different from belief revision.

More recently, Alberto was a central figure in the work on view-based querying. Starting with the innovative LMSS95 paper (Levy, Mendelzon, Sagiv, and Srivastava PODS 1995) that introduced the problem of answering queries using views, Alberto made several important contributions to the emerging area of view-based modeling and processing. His recent work on fine-grained access control (Rizvi, Mendelzon, Sudarshan, and Roy, SIGMOD 2004) explores an application of view-based reasoning to the important area of data security and access control. This paper is already on the must-read list in data security.

Alberto's research was central to the development of many areas of database research such as database design, semantic query optimization, graphical query languages, and querying web data. But this very impressive list is by no means exhaustive. He also made important contributions to recursive query languages, on-line analytic processing, similarity-base queries, data warehouses and view maintenance, algorithms for computing web page reputations, and indexing of XML data. Perhaps more than academic evidence, the personal testimonies remain as Alberto's most lasting legacy. Serge Abiteboul wrote:

> "Alberto belongs to a small group of people whose vision I trust. After many years, I realize how much I have learnt from discussions with Alberto, how essential his papers have been during these many years to shape the field."

Alberto was an active member of both the database theory and database systems communities. He served as the PC Chair for PODS in 1991, as General Chair in both 1997 and 1998, and on the PODS Executive Committee from 1997 to 2001. He served as PC Chair for VLDB in 1992, and as a member of the SIGMOD Executive Committee from 1998 to 2001. Jeff Ullman wrote that Alberto was instrumental in bringing the SIGMOD and PODS communities together. He helped to establish a system under which the two conferences were run as a single meeting with two independent program committees. The first joint meeting was in 1991. Jeff commented that this "appears to have worked out to everyone's benefit."

Alberto was a quiet man who did not seek out honors. He was modest about his role in shaping the foundations of relational databases and his pioneering work in laying the foundations for querying the web. His web page gives a glimpse into what Alberto found important. It prominently displays a 1978 picture of his colleagues and friends from Princeton whose friendship and trend-setting fashion sense he would refer to with a smile. Take a look at this picture. You will recognize many familiar faces of people who have gone on to shape computer science and database research – including a bearded and fully contented Alberto.

A few weeks before his passing, Alberto received the news that he had been elected to the Royal Society of Canada (the Canadian National Academy for Science, Engineering, and the Humanities). Alberto received the news with characteristic modesty. When it was suggested that he could now "rest on his laurels", he laughed and made it clear that he was not done yet!

In response to the news of his passing, condolence emails have flooded into Toronto from all over the world. Many of the messages came from South America. Alberto was instrumental in bringing many South Americans to the University of Toronto and his long list of graduate students, postdoctoral students, and visitors reads like a Who's-Who of South American academics. Alejandro Vaisman reports that in the early eighties, Alberto was a key contributor to the creation of the Computer Science Department at the University of Buenos Aires. Alberto Laender reports that Alberto was a good friend of the Brazilian database community. He visited Brazil several times and twice was the keynote speaker at the Brazilian Database Symposium.

Perhaps the most frequent theme in these messages were the appreciative remembrances of a kind, gracious, and welcoming Alberto who had helped and influenced many, many people, especially at the beginning of their careers. Claudia Bauzer Medeiros wrote:

> "Alberto was always ready to help students, and to promote the advancement of the careers of other people. I will forever remember him as someone who liked to laugh, was patient in his explanations, generous with his time and who respected others. A very friendly and charming person, full of joy and curiosity. It was a privilege to have known him."

Alberto's disease never stopped his work or dampened his humor. At the time of his passing, he was serving as the ICDT 2007 PC Chair and as a program committee member for ICDE 2006. He was also serving on the editorial boards of several journals, and as an associate editor for ACM TODS. Two of his papers were presented in September at VLDB 2005. A tribute to Alberto is being planned for PODS 2006.

We will remember a man who could talk appreciatively and knowingly about the latest Aerosmith or Coldplay album (he would of course cite J. M. Mendelzon and M. Mendelzon as the source of his knowledge on this and many other things). We will also remember Alberto's love of films, the Toronto Film Festival was one of his favorite yearly events. His partner Colette wrote:

> "Next to travelling [the film festival] was probably one of our favourite activities – not just the films themselves but the hanging out near hotels, hoping to spot movie stars, like a couple of very silly teenagers."

We cherished his fun-loving and teasing sense of humor, and the calm and open way he approached the various tasks and duties of academic life. Alberto was at his best after a good laugh, and he believed that innovation is more likely to flow from a research meeting at which there is laughter.

Alberto was the beloved partner of Colette Granger, devoted and proud father of José Manuel and Martin, stepfather to Emma and Paul, loving son of Maria Gloria Rabinovich de Mendelzon and the late José Mendelzon, dear brother of Daniel and Ricardo, and brother-in-law to Marta and Paula. He will be greatly missed by niece Laura, nephews Ariel, Guillermo and Andrés, uncles, aunts and cousins in Argentina and Paraguay, and many friends and colleagues around the world.

To conclude, some personal remembrances from friends.

"Alberto has been an internationally known leader in the development of database theory over the last 25 years. During these years, he continually produced original and significant research, which had tremendous influence in shaping this field. Alberto's work has had a profound impact on my own career. As a Master's student I read a paper he wrote as a Ph.D. student with Maier and Sagiv. I was greatly intrigued by the open questions they posed at the end of the paper, and these questions provided the inspiration for both my Master's thesis and Ph.D. dissertation. Alberto was not only tremendously respected by his colleagues for his scientific work, but was also extremely well liked for his unassuming manner, his sense of humor, and his warm personality. Even though we'd regularly meet only about once

or twice a year, I considered him not only a good colleague, but also a friend. I will miss him greatly." Moshe Vardi

"Alberto was an outstanding mentor and role model for the students he advised (a total of 17 PhD and 31 MSc students). Observing Alberto's interaction with his graduate students one concludes that providing effective guidance to students working through challenging research problems was the most relaxed and enjoyable of tasks. I benefited from his calm advice before the seemingly daunting task of giving your first research talk at a conference. Later on, I enjoyed seeing this ritual repeated with several of Alberto's later students. Alberto had an enjoyable sense of humor. It was hard to avoid giggling at one of his understated jokes: you had the impression that you were not supposed to laugh given Alberto's delivery in the most serious lecture tone that you could imagine." Mariano Consens

"I presented my very first PODS paper at the one run by Alberto in 1991. I was very young and green at that time, and initially quite afraid and not knowing what to expect of how things go at such a conference. But Alberto's relaxed presence put me readily at ease. After being introduced to him he immediately put me at ease. Alberto's presence will be dearly missed by the community." Jan Van den Bussche

"Alberto was a gem of a person, in addition to being a brilliant researcher. I will always cherish his memories. Alberto visited IIT Bombay during his recent sabbatical. It was a pleasure to see his enthusiasm to explore and learn. I remember him discussing several Indian movies he saw, preparatory to coming to India. It amused him no end to find one of them was set in Toronto and featured a shopping mall he used to visit!" S. Sudarshan

"In spite of living abroad for more than thirty years, Alberto has always maintained close ties with Argentina, particularly with Buenos Aires, a city he loved. He was a key contributor to the creation of the Computer Science Department at the University of Buenos Aires in the early eighties, giving talks and courses each time he visited the city. It was during one of these visits when I decided to start my PhD career. I will always be thankful to him for giving me the chance to be his student. Alberto's kind 'Welcome!!' each time I arrived at Toronto will forever remain on my mind." Alejandro Vaisman

"In 1988 Alberto gave a tutorial on 'Logic and Databases' at the International Conference of the Chilean Computer Science Society. I had just finished my PhD thesis on mathematical logic and was looking for research directions in computer science. Attending that tutorial was an illuminating experience and I knew that at some point I would go in that direction. In 2000 I invited him to Chile, where I had several students working with me on different subjects. I was impressed by the speed at which Alberto was able to interact with them, posing the right questions and giving them the right feedback. Alberto was always generous with his time, knowledge and ideas. The Latin American community, in particular, will miss him enormously." Leo Bertossi

"Alberto had a very unique personality. His clarity of thought and sharpness were and always will be a source of inspiration. He was often brief but his statements made you think deep. As a new Ph.D. student, I remember often getting excited about new papers that I just read and was asking him why we didn't pursue this or that. I found his response 'because we don't know that area well' very thoughtful. He was one of very few databases researchers who was working on the Web as early as 1994 and became the program chair of the WWW conference in 1999. Now that we can see the two areas (databases and the Web) have had so much influence on each other, his vision is admired. I enjoyed every moment of my Ph.D. work under his supervision and I will be missing him a lot." Davood Rafiei

"Alberto was a truly inspiring and incredibly likeable person. It puzzled me for a long time how such a busy and celebrated researcher could be so generous and patient in advising his students. With time, I witnessed the same dedication and competence in his teaching, either graduate or undergraduate courses, and in his service to the academic community. Alberto has taught me a lot more than his professional duties required, and has set an example that deeply inspired me. For his generosity, I am forever indebted. But, above all, I will greatly miss his constant good company, at a research meeting, at a technical session in a conference, or at a pub watching a soccer game." Denilson Barbosa

"My fond memories of Alberto are mostly personal, his wit, his humor, his love of movies. We always ended our academic discussions with: '... so have you seen any good movies lately?' Alberto and his Latin American students made me want to learn Spanish and I remembered how he corrected me when I used the Spanish plural form, which I thought was the proper form, to address him!" Dimitra Vista

"Alberto's research played a fundamental role in several quite different research areas, including relational database theory, belief revision, and Web querying. Many of his papers have become classics. For example, his paper with Hirofumi Katsuno 'On the Difference between Updating a Knowledge Base and Revising It,' that appeared in KR'91, is one of the most cited papers in the area of belief revision. However, Alberto was one of the true pillars of the SIGMOD/PODS community not only because of his many research contributions, his extensive service responsibilities, or his impressive record of supervising doctoral dissertations, but also because of his warm, friendly, and generous personality. He was genuinely interested in other people's research and took a pleasure in their achievements, and was always willing to help. He didn't shy away from criticisms but delivered them in a gentle, good-natured, and often humorous way. It is hard to believe that one will no more be able to stop by his office to discuss the latest research issues or just chat with him. We will all miss him very much." Jan Chomicki

"I do not remember when I first met Alberto. When I moved to Canada in 1984, he was already here. I think we got to know each other in the late 1980's. Starting with the 1992 VLDB Conference in Vancouver, we worked on a number of projects together. Alberto was always a source of calm, quiet and sound advice. In many ways, our personal-

ities were quite different, but I always enjoyed his company, his friendship and his understated humour. He was a force in the database community and his influence went way beyond the Canadian scene. I will miss him both as a friend and as a colleague." Tamer Özsu

"I will remember Alberto as a creative researcher who constantly pushed our field in new directions, a wonderful colleague and a generous supporter. Alberto impacted my career at several key points. As a graduate student, his paper on belief revision greatly affected the way I thought about formalizing the properties of irrelevance, the topic of my Ph.D thesis. A couple of years later, I actually met Alberto while he was visiting AT&T Bell Labs, and had the pleasure of working with him on answering queries using views. In addition to the productive collaboration, Alberto's feedback greatly contributed to strengthening my confidence as a young researcher. Later on, when my colleagues and I got interested in languages for querying semi-structured data, I found his work on WebSQL to be very influential on my thinking, and our StruQL and XML-QL languages were greatly impacted by it. In short, Alberto was one of few researchers whose work I always found interesting and inspiring, and whose support and friendship I could always count on. His presence in our community and his good humor in our informal gatherings will be sorely missed."
Alon Halevy

"I felt like a bigger person in Alberto's presence. There was an atmosphere of emancipation around him, which was very conducive to research and discovery. I wrote my best papers with Alberto as co-author, and cracked my best jokes in his company. There will never be anyone like him."
Gösta Grahne

"One oddball memory I have of Alberto is my visiting him while he was working at Yorktown Hts. It must have been in the very early 80's. Alberto showed me the new 'conversation pits' that they had installed in the hallways, and suggested we try one out. These were nothing but depressed areas with a whiteboard. So we started throwing symbols up on the whiteboard, talking about random topics about relational databases. A half hour later, we had the idea of acyclic schemas and the first of their interesting properties. We both talked the idea up, and the result was a paper with several other authors who had contributed properties — Beeri, Fagin, Maier, and Yannakakis." Jeff Ullman

"The database community owes a huge debt to Alberto, not only for his visionary research but also for his warm and cheerful personality through which he helped so many young researchers. Although I only met Alberto occasionally, I always regarded him as a friend, and we had plans to get together and do some joint work. Sadly this is not to be. Like all of my colleagues, I shall miss him greatly; but we can take some comfort in the inspiration and encouragement he has given to us." Peter Buneman

"It was the end of 1987 and I had just started my Ph.D. investigating the possible usage of diagrams and direct manipulation in interacting with databases, i.e., providing the user with a so-called visual query language. This idea is now almost trivial, but at that time it was quite new and intriguing. Basically, we had in mind database issues such as the expressive power of query languages, query evaluation, SQL, etc., and how to couple them with ease-of-use. While working on expressive power, I concentrated on expressing non-relational queries, such as transitive closure. Extending the expressive power of query languages was very fashionable at that time, especially because of the growth of Datalog, and graphical query languages with high expressive power were mainly proposed by Alberto Mendelzon's group at the University of Toronto (they were then working on the G^+ language). When I started studying their work, I was so impressed that I decided to concentrate much more on the formalization of QBD and compare its expressive power to G^+. We did not have e-mail (I know it is hard to believe, but this happened more than fifteen years ago) so I wrote a 'real' letter to Alberto. He replied quickly and in the meantime asked one of his Ph.D. students, namely Isabel Cruz, to give me more detailed answers, that helped a lot in writing my first journal paper. I was so impressed by the technical level of Alberto, I was still a young Ph.D. student and quite afraid of meeting an already famous researcher like Alberto. However, as soon as I got there everything was smooth, easy, and pleasant. Isabel and the other students were great, but Alberto was extremely friendly, as I would have never expected from a very busy professor. He spent a lot of time with us, both discussing visual languages and just enjoying. After that visit we met many other times, and he was one of those rare people who is always a pleasure to meet. His jokes and smiles as well as his scientific work will stay with us forever." Tiziana Catarci

"What to remember about this wonderful man? In a desperate search for the 'right thing to say,' I dug into the tons of emails we've exchanged over the years. His comments, elegant proofs, very challenging questions, the usual business from a great mind... But, in the middle of the draft of a grant proposal, you will find some (so very to the point) Bob Dylan lyrics. And beautiful extracts from poems by Cummings and Ferlighetti. A referee report prefixed by his brilliant critic of a new movie...

This is how I remember Alberto, a wonderful colleague, so smart and so warm, who loved research and, with just the same enthusiasm, loved life... and made you love both."
Tova Milo

Outside the leaves were falling
and they cried
Too soon! Too soon!
Ferlinghetti

"Alberto was since the very inception of the Web interested in the data models and database problems arising from this new phenomena of information management. He worked on XML databases and query languages for the web, and pioneered research on databases in the RDF data model, the recommendation for a metadata model for the Web of the W3C. His last PODS paper (2004) introduces normal forms and studies their complexity for RDF data as well as the theoretical basis of conjunctive queries over this data. This work established the connection between database theory and semantic web research."
Claudio Gutiérrez and Carlos Hurtado

"I started working with Alberto on relational theory in grad school at Princeton in 1977. My thesis work was on NP-completeness of sequence problems, but Catriel Beeri had arrived in 1976 and gotten Jeff Ullman and most of his students started doing database theory. I started working with Alberto and Shuky Sagiv on that topic, too, while writing up my thesis. Alberto often served as my personal tutor in the area, and our collaboration continued while he was at Watson and I taught at Stony Brook. Looking back, I see he was the Princeton classmate I wrote most papers with. While the direct collaboration wound down after he moved to Toronto, we still enjoyed interacting, which included a summer he and his family spent here in Oregon in the 1980s. I feel like a little piece of my history has passed away with him; I'll miss him." David Maier

"My first 'contact' with Alberto came when I began reading his research on dependency theory and on the fundamental connections between properties of hypergraphs and database design. His pioneering work was inspirational in shaping my own doctoral work. Interestingly, I later became Alberto's postdoc! Indeed, I was one of his earliest postdocs. I was delighted when I received the news of the postdoc offer, conveyed to my Alma Mater, Indian Institute of Science, Bangalore, and relayed by my friends to me at a remote village, where I was vacationing with my parents. Once I arrived in Toronto, there were so many things new and unfamiliar to me. I still vividly remember the warm and supportive environment that Alberto offered me. In fact, it was Alberto who decided 'Lakshmanan' is too complicated for the North American tongue and 'officially' made 'Laks' my first name. Alberto's incisive, methodical, and no-nonsense approach to research would leave an indelible impression on me for years to come. It is a pleasure and privilege to have known this great man, who faced his disease and his ultimate death with the same calm, courage, and repose that he displayed when attacking fundamental problems in database theory. I will miss him dearly."
Laks V.S. Lakshmanan

"In the same way as many of our colleagues, I first 'encountered' Alberto by reading one of his papers. For me, it was the STOC 1979 paper on schema equivalence, often cited as BMSU, and then the 1979 MMSU papers on adequacy of decompositions. I was working on my 'laurea' thesis at the time, and all the authors were impressive to me, by definition. A few years later, Alberto would show me his great personality, humor and modesty by just saying 'yes, I am the "M" in these papers.' I am deeeply indebted to Alberto: I went to Toronto as a postdoc, formally visiting another group, but he invited me to collaborate with his as well, and in the subsequent years he hosted students of mine for midterm visits, which were really productive, as he was always open to advise, listen and collaborate. I also had the pleasure to host him in Rome various times, and we could share long discussions on many different things, not only scientific. We also shared interest in soccer, and we would often comment on it, especially with reference to the many Argentinian and Brazilian players in Italy. We missed soccer in Toronto, and when he took me to a baseball game as a surrogate, I promised that during his next visit to Rome, I would take him to a game at the Olympic Stadium—I wish I could still keep my word." Paolo Atzeni

"I knew Alberto when he lectured a short course on Logic and Databases in 1987, at ESLAI, La Plata, Argentina. At that time I was a part time lecturer on databases, mainly on the practical side of them, while still working in the industry. In his lectures I discovered a fascinating new dimension in the field, which made me eager to learn about logic and theory in general. It was a big effort at that time for me to pass his course, given my lack of theoretical formation. But that course, and the few talks I had with Alberto during it, meant something very important for my life. I think that it was at that point that I decided to move from the industry to academics. I started to study on my own and finally, in late 1990, at a rather unusual age of 38, I decided that I should do a PhD. Of course I thought of Alberto, and he kindly accepted. That was extremely important to me. He gave me his confidence and support in so many ways, and our meetings where so illuminating to me, that I am convinced at this point that it was him who made it possible for me to carry on with the change I expected for my life. He also very generously assisted me with his funding for a visit to him for six months in 1995. That period of time was essential for my research. Alberto was not only a brilliant and successful scientist and teacher, he was mainly an excellent human being, and a part inside me died together with him."
José María Turull-Torres

"I remember once mentioning to Alberto that he had been my PhD advisor. With his characteristic and unforgettable smile he corrected me and said that the past tense was not appropriate as he still was my PhD advisor. I also remember well the long research meetings in his office, where he never failed to impress me with his quick and critical thinking. I was particularly mesmerized that he always seemed to be at the center of some new and exciting result. Other more informal conversations we had over the years would flow effortlessly between topics, including the appreciation of a good wine or of music (true to his origins, Alberto was proud of Argentinian musicians and of sharing his last name with Felix Mendelssohn's). I remember particularly two recent dinners where Colette was also present, one in Chicago in 2002 (right after SIGMOD) and one just last year in Toronto (during VLDB). The former dinner was particularly upbeat as we had plenty to celebrate: his upcoming sabbatical (he was making plans to come to Chicago for a couple of months) and my new job. However, last year's dinner was particularly moving as we reminisced about the 'old times' and especially as Alberto acknowledged ever so discreetly the loving support of Colette and of his two sons Manuel (also present) and Martin. I will cherish these and many other fond memories I have of Alberto Mendelzon, the brilliant researcher and caring mentor who will always be my PhD advisor." Isabel Cruz

"Alberto Mendelzon was my advisor during my PhD years in Toronto. Though, to say that he was merely my academic advisor, would be an understatement. Alberto definitely played a major role in shaping my way of thinking about research problems and in determining what are the right questions to ask when trying to tackle them. I feel extremely fortunate for having him as my guide in this endeavor.

Nevertheless, outside of his academic suit, Alberto was equally approachable, engaging, and fascinating. He was ready to get involved in stimulating discussions on a wide

variety of topics, and eager to learn about anything that escaped his broad knowledge. He had a certain predilection for Greek easter cookies, and was passionate about blues.

I think that Alberto had always been flirting with realismo magico. Thus, he would sprinkle his life with both salt and pepper. He would take joy from the small details of life, and he would never be anything less than generous. I also believe that he would be very content to know how much, and in how many different aspects, he managed to positively influence the lives of the people around him. I can almost picture him now: smiling back to us, with one of his wide, modest, heartfelt smiles...

Hasta luego, maestro!" Themis Palpanas

Edited by Renée J. Miller and the Toronto Database Group

Tips on Giving a Good Demo

Mary Fernández
SIGMOD 2005 Demonstrations Chair
mff@research.att.com

For the first time this year, a "Best Demonstrations" session was included in the SIGMOD program. The first two days of the program included 24 demonstrations, each of which was presented during two of six interactive demo sessions. During the first two days, panels of three judges visited each demo group, each of whom was allotted 15 minutes to present their system to the judges. The friendly competition made for very exciting and noisy demo sessions!

The judges selected three demonstrations [1, 2, 3] as the "best" based on five criteria that we share with you here. The selected demos were showcased in front of an audience in a special session on the final day of the conference.

Giving a good demonstration is like giving a good "elevator talk", but it is presented interactively. If you are unfamiliar with the classic elevator talk, here is the demonstration variant: Imagine that you get stuck in an elevator with your favorite high-tech mogul (e.g., Gates, Jobs, Ellison, Page & Brin), and you have fifteen minutes to sell your new database systems technology. You start to describe your idea, and the mogul says, "Don't tell me! Show me!". So you unsuspend your laptop (you are always prepared), and start to present your demonstration system.

Since giving a good demo is like telling a good story, we use a literary metaphor to describe each criteria. Do not underestimate the importance of telling a compelling story, because it helps people better remember your technical contribution.

User scenario : The characters

Introducing the users, or characters, in your story answers the key questions: *Who* is your target user and *why* are they important?

Your demonstration should answer these questions by describing a compelling user scenario or experience. If you are going to work very hard to build a system, it is likely that you have a particular user in mind. Database systems have numerous potential users: from end-users at their desktop, to database administrators, to analysts and statisticians, to animators and gamers—the list goes on and on. Know your user and put them at the center of your demo story.

Technical problem : The setting

Introducing the technical problem, or setting, answers the key question: *Why* does my system exist?

Your demonstration should answer this question by identifying the core technical problem. Ideally, your system should demonstrate the problem *interactively*. For example, if your system implements a query optimizer, you could demonstrate the problem by running queries without the optimization enabled, let the user see how slow it is, then possibly present a graph of the unoptimized query speed. As another example, if your system implements a graphical/high-level interface to a query language, you could show the queries that a user might have to write by hand in the absence of your interface.

If you cannot find a way to demonstrate the technical problem interactively, an effective alternative is to describe the problem graphically in a poster or in slides.

Technical solution : The plot

Presenting the technical solution, or plot, answers the key questions: *What* does my demonstration system do and *how* does it work?

Demonstrating your core technical solution is the centerpiece of your presentation. Ideally, your demo should include both the system's "dashboard" (i.e., its user interface) and a look "under its hood" (i.e., its internals). You should now return to your user scenario and show how it is handled using your system's dashboard.

Demonstrating the internal functionality of your system may be more difficult. Your technical solution, for example, may be an algorithm embedded deeply inside a query engine. Some effective techniques from the 2005 demo program included dynamic graphs that plotted relevant system variables such as CPU load and locations of distributed query plans; output of each phase in a query translation process; and displaying the intermediate representations of mediated data sources.

Making this internal functionality visible externally may require "extra" work that would not otherwise be necessary in a real system, as users typically only care about the system's dashboard, not what is going on under the hood. Visitors to your demo, however, may be more interested in the internals, so demonstrating the internals interactively will help visitors remember your technical contributions.

If your system solves multiple problems, an effective presentation technique is to introduce one problem at a time (using poster or slides), immediately followed by a demo that shows its solution. This technique draws the visitor into your demo quickly and avoids the common mistake of using all your time to describe the problems and not having sufficient time to show your system.

Integration : The sub-plots

Describing how your system is integrated with other technologies, or sub-plots, answers the key questions: *What* are the components of my demo system and *how* does my technical solution, interact with, support, or enhance related technologies?

The most interesting database systems are part of integrated systems. You should clearly identify the components of your demonstration system that you invented and engi-

neered entirely yourself, those that were integrated without any changes, and those that were modified to integrate with your technology. Some examples from the 2005 demo program included an open-source, 3-D game that was re-engineered to support interactive, streaming queries of character location and action, and a personal information system that used a desktop file metaphor to browse inferred relationships between heterogeneous data sources.

If your technology integrates easily or seamlessly with existing technologies, that is a sign of good design and engineering. You can illustrate this aspect, for example, by showing how much code had to be changed to integrate the components.

Impact : The resolution and insight

Describing the potential impact of your database systems research, or the resolution of your story, answers the key question: *How* and *when* might my technology have an impact?

If you are lucky, you might have one or two minutes to try to put your demo system in perspective. Making a guess about how your own work or similar systems might change the world may seem speculative and a bit egotistical, but informed speculation is very interesting to your visitors and helps them relate your technical contribution to what they already know.

Lastly, this is a good time to talk about the lessons that you learned. Often we start building a system with assumptions about what the "hard" problems are only to discover that the genuinely hard problems are something entirely different. Experiential knowledge of this kind is valuable and rarely shared in written form, but also of great interest to your visitors.

Pointers on Demo Form

Your demonstration will have a bigger impact if your visitors can see and hear you! Here are a few pointers on form to improve your presentation.

- *All* your visuals (posters, displays, slides etc.) should be legible to your visitors, who may have to stand 5-10 feet away from the display. Fonts should be at least 20 point.
- Bring or rent a monitor or video projector so that many people can view the demo at the same time. Hanging over a laptop is uncomfortable for everyone involved and limits the number of visitors at any one time.
- A poster is an advertisement for your demo, so it should be visually interesting. One or two large posters is more effective than hard copy print-outs of slides hung on the wall or an easel. If you prefer slides, present them on your laptop.
- *Point* directly to your visuals. Gesturing at a poster, slide, or display that is 10 feet away is ineffective and confusing. If you use a poster, stand next to it and point directly to the poster while describing the user scenario and technical problem.
- Demo rooms are *noisy*, so ask visitors to stand close to you so they can hear you. Speak directly to them, not at the poster or laptop.
- *Practice* your demonstration, more than once. Visitors will ask questions and things will go wrong, so do not expect to get through your entire demonstration every time. Don't leave the most important points to the end, because you may not get to them!

Putting it all together

You may find all these suggestions overwhelming and difficult to satisfy simultaneously, but consider them as opportunities to improve your communication skills. Learning research[1] has shown that different people learn best through different modalities: verbal (say it!), aural (hear it!), visual (see it!), and tactile (touch it!). Every person has a modality in which they best express themselves and in which they learn most effectively. Demonstrations are an excellent way to develop skills in multiple modalities, because they can incorporate many forms of communication. Giving demos will not only make you a better communicator, but will make your work accessible to more people.

Giving a demo is an immediate, intimate, and exciting way to share your technical work with others. We hope that you find these suggestions helpful in giving demonstrations of your database systems research.

References

[1] Y. Ahmad, B. Berg, U. Cetintemel, et al. Distributed Operation in the Borealis Stream Processing Engine. In *SIGMOD*, Demonstrations Program, June 2005.

[2] Y. Cai, X. Dong, A. Halevy, J. Liu, J. Madhavan. Personal Information Management with SEMEX In *SIGMOD*, Demonstrations Program, June 2005.

[3] Y. Li, H. Yang, H.V. Jagadish. NaLIX: An Interactive Natural Language Interface for Querying XML. In *SIGMOD*, Demonstrations Program, June 2005.

[1]Howard Gardner, Harvard University, http://www.howardgardner.com/

Information Source Selection for Resource Constrained Environments

Demet Aksoy
Computer Science Dept.
One Shields Ave.
University of California, Davis
530-752-3601
aksoy@cs.ucdavis.edu

ABSTRACT

Distributed information retrieval has pressing scalability concerns due to the growing number of independent sources of on-line data and the emerging applications. A promising solution to distributed retrieval is metasearching, which dispatches a user's query to multiple sources and gathers the results into a single result set. An important component of metasearching is selecting the set of information sources most likely to provide relevant documents. Recent research has focused on how to obtain statistics for the selection task. In this paper we discuss different information source selection approaches and their applicability for resource-constrained sensor network applications.

1. INTRODUCTION

Finding information is increasingly difficult due to the explosion of content that has resulted from advances in computer networking and data storage. Widely distributed information makes it difficult to issue a single query and get globally optimal results. Even in traditional applications of digital libraries such as the one offered by the University of California at Davis [19], there are numerous electronic databases that can be searched. A user must select which databases are most relevant, issue a query to each of those databases, and then review all of the result sets to retrieve the top matching documents. Similar problems arise in the case of information retrieval from a federation of sensor networks.

Some sensor networks make the collected data publicly available on the World Wide Web for general use [6]. In this case, even large general search engines fail to thoroughly analyze the growing content [12]. Therefore there is a new trend towards specialized search engines in particular topics. In addition, beyond the issue of scalability, many sites provide dynamic content that can not be accessed by the hyperlink traversal strategy used by web crawlers. For instance, consider a keyword search page that provides customized results to the user; a web crawler will only consider the text on the search page as a document, but does not have hyperlinks to the documents available beyond this facade. The non-crawlable content becomes part of what is commonly known as the *Hidden Web*. Similar issues arise in sensor network queries.

One approach to querying distributed content is to dispatch the query to each information source that is likely to contain the requested documents and merge the query results before presenting them to the user. This approach, known as metasearching, has received considerable research attention, e.g., [2, 6, 10, 11, 17, 18]. Selecting information sources that are likely to contain the requested document is the first key for a successful search

This paper focuses on *information source selection* for a metasearching system for applications where system resource limitations and restrictions need to be taken into consideration. The paper is organized as follows. Section 2 describes the design of a general metasearch engine and discusses the issues that arise with the introduction of resource constraints. Section 3 presents various approaches for information source selection for metasearching. Finally conclusions and future research directions are presented in section 4.

2. METASEARCH ENGINES

The main challenges in implementing a successful metasearch engine can be categorized as (1) *information source selection*: the process of determining which information sources are likely to contain relevant content, (2) *query execution*: the process of sending the user's query to each of the selected information source, and (3) *result merging*: the process of aggregating the results from each of the selected source and presenting the final result to the user.

The metasearch engine needs to start with a good classification of the information sources it receives results from. Due to the amount of data and the number of hosts available, and the time limitation of power limited devices efficient information source selection needs special attention. The retrieval results usually need to be filtered and narrowed down before presented to the user. Context

and location are two important factors for such filtering. To provide a good filtering criteria, one needs to ensure that the initial set was comphrehensive and accurate.

The simple approach of sending queries to *all* information sources is not scalable to the large number of hosts and large collections. An effective information source selection algorithm must efficiently use resources while maintaining a low probability that relevant documents are missed. Emerging applications of sensor networks have additional challenges due to resource limitations in terms of energy, storage, and processing capabilities. For instance, limited power of mobile devices and their unpredictable environments require that the querying be efficient and information retrieval results arrive fast. Otherwise the clients will need to consume considerable time in idle listening mode which was shown to be significantly high [13]. In general, data generated by a federation of sensors can be quite dynamic and change frequently. This requires an adaptive approach.

Our paper focuses on information source selection for metasearching in a widely distributed environment with heterogeneous sensor network federations. In the following section we present various information source selection algorithms and provide insights for how federated sensor information retrieval applications can be supported.

3. COMPARISON OF INFORMATION SOURCE SELECTION APPROACHES

Research in information source selection was originally centered on the assumption that domain administrators export a standardized and accurate content summary for each information source. In traditional database applications, a content summary for a database consists of a *term frequency vector* and a *document frequency vector*. The frequency vector counts the number of times each unique term appears over all documents in the database. The document frequency vector counts the number of unique documents that contain each term. Database selection algorithms that require the export of content summaries are called *cooperative* database algorithms, e.g., CORI [5], GlOSS [9], and CVV [23]. Such algorithms presume that the database administrators *can* export the content summary, that they *want* to export it, and that they *will* do it accurately. In an open system, these assumptions are unrealistic for two main reasons. First, information providers may have an incentive to skew the content summaries in favor of their site. Second, the content summaries are difficult to standardize without detailed communication on semantics such as suffix stemming and stop words [8].

An alternative approach is taken by [2, 4, 11, 21]. These algorithms obtain information by issuing queries to the information source and receiving document counts and/or document samples. Such approaches are geared towards widely distributed collections with multiple administrative domains. In the following, these approaches are described in two main categories: query sampling and topical content summaries.

3.1 Query Sampling

Query based approaches can be categorized between those that obtain information on-demand for each incoming query, and those that maintain persistent information about each information source. On-demand approaches execute an initial probing query on all information sources in such a way as to minimize the computational cost on the servers. Preliminary query results are used to select a subset of the information sources for further execution of the original user query. Persistent approaches gather information from each information source before any user queries are processed.

3.1.1 On-Demand Query Sampling

In Hawking et al's study [12], the idea of "lightweight probes" is introduced. These are queries that are handled differently than normal queries. In response to such probes, the server is expected to reply with the following information: 1) the total number of documents on the server, 2) the number of documents containing a specified number of the query terms within a specified proximity of each other, 3) the number of documents in which a specified number of the query terms co-occur, and 4) the number of documents containing each individual query term *t*.

The collected information is used to estimate the number of relevant documents and to allow the metasearcher to select a subset of the information sources to execute the full query. The main problem with this approach is that it requires the cooperation of servers to process lightweight probes in a different manner than standard queries. This problem is addressed in [2] which takes the approach of executing the full query on each information source. To reduce the costs of query execution, only the first few records are retrieved from each information source. These initial results are evaluated to determine which information sources are most likely to contain relevant documents. Though this approach solves the non-standard lightweight query probe requirements of [12], it introduces a new problem as will be discussed at the end of this section.

3.1.2 Persistent Query Sampling

Voorhees et al. [20] studied the persistence of information source statistics retrieved via query sampling. The approach begins by considering the theoretical optimum performance of a result aggregation algorithm. The problem can be stated as follows: given a query Q, information servers $I_1, I_2, \ldots, I_L$ and N, the total number of documents to be retrieved, find the values of $\bullet_1, \bullet_2, \ldots, \bullet_L$ such that

$$\sum_{i=1}^{L} \bullet_i = N \text{ and}$$

$$\sum_{i=1}^{L} F_Q^I(\bullet_i) \text{ is maximum.}$$

where $F_Q^I(\bullet)$ is the number of relevant documents retrieved as a function of total retrieved documents. The challenge is to find a reasonable approximation for this number so that this optimization can be performed. The authors propose that the relevant document recall characteristics of two queries will be similar so long as the queries themselves are topically similar. Therefore, if recall characteristics are stored for a set of training queries, then the recall characteristics of later queries can be estimated by finding queries similar to the ones issued during the training phase. The topical similarity of queries is determined by clustering the training queries and then comparing the incoming query with the cluster centroid using cosine similarity. The main problem with this approach is that computing the recall characteristics for each training query requires knowing which records are relevant. Such an approach works well in a test environment that includes relevance judgments. It does not, however, scale to an environment with many information sources, many topics, and no prior relevance judgments. The manual effort involved in evaluating document relevance is impractical.

The idea of creating estimated content summaries using random sampling has been investigated in [4]. The algorithm is as follows:

1) Select an initial query term.
2) Run a one-term query on the database.
3) Retrieve the top N documents returned by the database.
4) Update the content summary based on the characteristics of the retrieved documents.
5) If the stopping criterion has not been reached, select a new term and go to step 2.

The initial query term is randomly selected from a general dictionary and two stopping criteria are provided. The first one is to stop sampling when a fixed number of documents are retrieved (e.g., n=300). The second one is a more complex metric that examines the change in the rank of terms during periodic points in the sampling. The stopping criteria are met if the change in ranks drops to below a minimum value.

The estimated content summaries can be used with standard cooperative database selection algorithms. Callan et al. [4] analyze the effects of estimated summaries on the performance of CORI [5]. Powell et al. [16] continued this work by examining the effects on GlOSS [9] and CVV [23]. These experiments show that the estimated content summaries are accurate enough for use with CORI, though there is performance degradation with GlOSS and CVV.

Si et al. [18] investigated the use of random sampling for a database selection algorithm that uses a language modeling approach. Language modeling considers information retrieval from a natural language processing perspective. Rather than computing the probability that a document matches a query, it computes the probability that a query is generated from the language model of a document [15]. That is, given the distribution of terms in a document, it computes the chance that randomly selecting terms from this distribution will result in the generation of the given query. [18] shows that this language modeling approach when used with random query sampling yields better performance than CORI.

Gravano et al. [11] propose that query sampling can be improved by creating a hierarchical classification tree to focus the queries on increasingly specific topics. The classification tree is generated during a training phase by a document classifier that creates rules for each topic using sample documents. The particular classification tree chosen in [11] was the first few levels of the Yahoo! hierarchy. The terms for each rule are sent as queries to the database and the number of matching documents is recorded. Two thresholds are set to decide whether a database should be assigned to a topic. The first one, *Coverage*(t_i), is the absolute number of documents in the database that match the query terms for topic t_i. The second one, *Specificity*(t_i), is the fraction of documents that match the query. As the classification algorithm issues the queries, the content summaries for each database are built up by scanning the terms in the retrieved documents and adding their frequency. The actual document count for single term queries are used as anchors to estimate the document count for those words that were not explicitly queried with a single term query. The content summaries of the databases are summed together

based on their position within the classification tree. This turns each node in the tree into a super information source that contains all of the information sources underneath it. When a user issues a query, the query terms are first checked against the content summaries at the top of the tree. The process then compares the query with the topics on the next level of the classification. This continues until the desired number of databases is identified, at which point the queries are issued directly to those databases. The details of the document classifier are presented in [10].

The experiments in [10] suggest an improvement in the content summary accuracy over random document sampling for a fixed number of samples, as well as an improvement in overall retrieval performance. However, the need for supervised machine learning (the document classifier) is a major weakness because obtaining high quality training documents for a non-trivial topic hierarchy is unrealistic. Another disadvantage is the reliance on accurate document counts from the server, which the administrator may skew to make the database more likely to be selected.

A similar approach to [11] is taken by Wang et al. [21] that does not rely on accurate document counts. However, it still requires a manually constructed topic tree, as well as a description for each topic, rather than document samples for each topic. Gravano et al. [10] show that the classification performance of [21] is worse than the earlier approach.

3.2 Topical Content Summaries

The results of [20] and [11] suggest that estimating the quantity of topically related sets of documents in an information source can improve selection performance. In addition to [20] and [11], two other studies using term distributions to model topics are [22] and [17].

In [22], an unsupervised clustering algorithm is run against all of the documents in a distributed collection. Rather than distilling the contents of the database into a single content summary, a term vector is computed for each topical cluster. Incoming queries are compared to the term vectors, and the closest matching clusters are queried for matching documents. The results from the experiments show that the retrieval performance approaches the same level as centralized retrieval. The main weaknesses in this algorithm is the need for each database to export a set of topical content summaries, and the need to limit the queries to documents within a selected set of topics. Both of these requirements entail a substantial amount of cooperation from the database. Shen et al. [17] take a similar approach as [22], where the main difference is a more complex method of clustering documents and comparing incoming queries to the clusters.

For such topical approaches to be practical outside of a test environment, it is necessary to generate the topical content summaries without the need for a cooperative environment. The existing sampling solutions are inadequate for this task. The random sampling approach of Callan et al. [4] fails to provide data on topics when the sample size is small, the database is large, and the topics are not uniformly distributed. A simple example best illustrates this. Assume we have a database D_1 containing 10,000 records, where 100 documents $d_{1,T}$ are related to topic T. The probability of randomly selecting a document related to topic T is given by:

$$p_{T,1} = \frac{|d_{T,1}|}{|D_1|} = \frac{100}{10{,}000} = 0.01$$

Further assume that we have a second database D_2 containing one million records where 100 documents are related to topic T. The probability of randomly selecting a document $d_{2,T}$ related to topic T is given by:

$$p_{T,2} = \frac{|d_{T,2}|}{|D_2|} = \frac{100}{1{,}000{,}000} = 0.0001$$

The number of documents, n, that must be randomly sampled to give a 95% probability p_c of detecting topic T in database D_1 is given by:

$$(1 - p_c) = (1 - p_T)^n$$
$$\log(1 - p_c) = n \log(1 - p_T)$$
$$n = \frac{\log(1 - p_c)}{\log(1 - p_T)}$$

Based on this, we would need to retrieve approximately n = 300 documents to have a 95% chance of detecting topic T in database D_1. On the other hand, 30,000 documents are required to have a 95% chance of detecting topic T in database D_2 because the probability $p_{T,2}$ is much smaller than $p_{T,1}$. To ensure a high probability p_c that all topics are represented in the content summaries, the number of documents required by random sampling is two orders of magnitude greater for database D_2 than D_1. In fact, the number of document samples required grows linearly with the size of the collection. It is therefore not a scalable solution unless one also assumes that the number of documents per topic also grows linearly.

Table 1. A comparison for different information source selection approaches

Technique	*Advantages*	*Disadvantages*
On-Demand Sampling	Can capture the latest information stored based on a particular query	High overhead, initial query time can not be masked
Persistent Query Sampling	Relatively low overhead	Difficult to produce the framework that can provide a good sampling
Topical Content Summaries	Provides detailed information based on individual topics.	Requires a good categorization of topics

The approaches described in [20], [21], and [11] do not require a linear increase in the number of samples to detect topics, but they introduce the problem of manual training. [20] requires a prior relevance judgment for each training document. Wang et al. [21] require a pre-constructed topic hierarchy with a textual description of each topic. Gravano et al. [11] also require a pre-constructed topic hierarchy, as well as document samples for each topic so that the document classifier can generate query terms. All of these approaches are not practical outside of a test environment due to the manual effort involved.

Lawrence and Giles [14] estimate that the largest search engines cover less than 20% of the total number of documents on the World Wide Web. However, when the documents managed by all of the search engines are combined, the overall coverage increases dramatically. The effect of this increased coverage in conjunction with an accurate database selection algorithm is investigated by Craswell et al. [6]. In addition to these inherent problems the associated increase in multimedia content in databases, broadcasts, streaming media etc. has generated further requirements for more effective access to giant global information repositories for mobile clients.

The definition of a topic however is ambiguous. Different users may choose to classify documents into different topics. In addition, the same user may choose to assign a document to different categories depending on the current information need. Despite these problems, simple approaches that model topics based on term distributions appear to be successful.

3.3 Comparison

In Table 1 we present a comparison of various approaches in three major categories. As suggested Persistent Query Sampling approaches are not appropriate for dynamically changing contents in the information source. On-Demand Sampling, on the other hand, results in a high latency and resource usage since the generation of the first records result in the most significant latency.

An interesting question is how these approaches can be applied in a sensor network environment where different administrations make their data publicly available. If the base stations of sensor networks are directly connected to the Internet, we result in a similar setting to traditional information retrieval within the Internet. The main difference would be the dynamically changing content with updates from the sensor nodes even though the type of the information that is published stays static in most cases. In this regard, topical Content Summaries when combined with approximations of On-Demand Sampling can provide a good trade-off for federated sensors.

Retrieving information directly from the sensor nodes deployed in the field, on the other hand, requires additional power, space and storage optimizations. Due to the resource constraints of sensor networks, none of these approaches can be directly applied in this emerging field. It is important to adapt information source selection approaches to provide a trade-off between efficiency and accuracy [1]. A key requirement for information source selection algorithms is to obtain topical content summaries in non-cooperative environments using a scalable number of document samples and without the need for manual training.

4. CONCLUDING REMARKS

Emerging applications such as sensor networks have limited resources. In this paper we provided a survey of query sampling based methods for obtaining information source selection statistics. The on-demand approaches either require a cooperative environment or introduce latency and excessive resource usage. Of the persistent approaches, only random sampling is practical due to the manual training requirements of the other solutions. Though random sampling provides accurate estimated content summaries for selection algorithms, it is not a scalable solution for algorithms that require the identification of topics. On the other hand topical selection algorithms outperform single content summary approaches, and therefore represents an avenue for further

research for research constrained environments of sensor networks.

5. REFERENCES

[1] D. Aksoy, M.S. Leung. *Pull vs Push: A Quantitative Comparison for Data Broadcast*. In Proc. of IEEE GLOBECOM, 2004.

[2] F.Abbaci, J. Savoy, and M. Beigbeder. A Methodology for Collection Selection in Heterogeneous Contexts. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC)*, Las Vegas, NV, pages 529-535, 2002.

[3] J. Callan and M. Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems (TOIS)*, 19(2), pages 97-130, April 2001.

[4] J. Callan, M. Connell, and A. Du. Automatic discovery of language models for text databases. In *Proceedings of the International ACM Conference on Management of Data (SIGMOD)*, Philadelphia, PA, pages 479-490, 1999.

[5] J. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *Proceedings of the Eighteenth International ACM Conference on Research and Development in Information Retrieval (SIGIR)*, Seattle, WA, pages 21-29, July 1995.

[6] N. Craswell, P. Bailey, and D. Hawking. Server selection on the World Wide Web. In *Proceedings of the Fifth ACM Conference on Digital Libraries (DL)*, San Antonio, TX, pages 37-46, 2000.

[7] K. A. Delin, S. P. Jackson, S. C. Burleigh, D. W. Johnson, R. R. Woodrow, and J. T. Britton, The JPL Sensor Webs Project: Fielded Technology , In *Proceedings of Space Mission Challenges for Information Technology*, Pasadena, CA, July 2003.

[8] L. Gravano, K. C. Chang, H. Garcia-Molina, and A. Paepcke. STARTS: Stanford proposal for Internet Meta-Searching. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, Tucson, AZ, pages 207-218, May 1997.

[9] L. Gravano, H. Garcia-Molina, and A. Tomasic. GlOSS: text-source discovery over the Internet. *ACM Transactions on Information Systems (TOIS)*, 24(2), pages 229-264, June 1999.

[10] L. Gravano, P. Ipeirotis, and M. Sahami. QProber: A system for automatic classification of hidden-web databases. *ACM Transactions on Information Systems (TOIS)*, 21(1), pages 1-41, January 2003.

[11] L. Gravano, and P. Ipeirotis. Distributed Search over the Hidden Web: Hierarchical Database Sampling and Selection. In *Proceedings of the Twenty-Eighth VLDB Conference*, Hong Kong, 2002.

[12] D. Hawking, and P. Thistlewaite. Methods for information server selection. *ACM Transactions on Information Systems (TOIS)*, 17(1), pages 40-76, January 1999.

[13] B. Hohlt, L. Doherty, and E. Brewer. *Flexible Power Scheduling for Sensor Networks*. IPSN 2004.

[14] S. Lawrence and C. L. Giles. Accessibility of information on the web. *Nature*, 400, pages 107-109, 1999.

[15] J. Ponte and W. Croft. A language modeling approach to information retrieval. In *Proceedings of the International ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 275-281, 1998.

[16] A.L. Powell, J.C. French, J. Callan, M. Connell, and C.L. Viles. The impact of database selection on distributed searching. In *Proceedings of the Twenty-Third International ACM Conference on Research and Development in Information Retrieval (SIGIR)*, Athens, Greece, pages 232-239, 2000.

[17] Y. Shen, and D. Lee. A Meta-Search Method Reinforced by Descriptors of Clusters. In *Proceedings of the Second International Conference on Web Information Systems Engineering*, Kyoto, Japan, pages 129-136, Dec 2001.

[18] L. Si, R. Jin, J. Callan, and P. Ogilvie. Language modeling framework for resource selection and results merging. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management (CIKM)*, McLean, VA, pages 391-397, 2002.

[19] University of California Davis Digital Library, 2005.

[20] E. Voorhees, N. Gupta, B. Johnson-Laird. *Learning collection fusion strategies*, In Proceedings of the Eighteenth International ACM Conference on Research and Development in Information Retrieval (SIGIR), Seattle, Washington, pages 172-179, July 1995.

[21] W. Wang, W. Meng, and C. Yu. Concept hierarchy based text database categorization in a metasearch engine environment. In *Proceedings of the First International Conference on Web Information Systems Engineering*, Hong Kong, pages 283-290, June 2000.

[22] J. Xu and W. B. Croft. Cluster-based language models for distributed retrieval. In *Proceedings of the Twenty-Second International ACM Conference on Research and Development in Information Retrieval (SIGIR)*, Berkeley, CA, pages 254-261, 1999.

[23] B. Yuwono and D. Lee. Server ranking for distributed text retrieval systems on the internet. In *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 41–49, Melbourne, April 1997.

LiXQuery: A Formal Foundation for XQuery Research

Jan Hidders Philippe Michiels* Jan Paredaens Roel Vercammen*

University of Antwerp
Middelheimlaan 1
B-2020 Antwerp, Belgium
{jan.hidders, philippe.michiels, jan.paredaens,roel.vercammen}@ua.ac.be

ABSTRACT

XQuery is considered to become the standard query language for XML documents. However, the complete XQuery syntax and semantics seem too complicated for research and educational purposes. By defining a concise backwards compatible subset of XQuery with a complete formal description, we provide a practical foundation for XQuery research. We pay special attention to usability by supporting the most typical XQuery expressions.

1. MOTIVATION

XQuery's popularity for querying XML documents is growing rapidly[1, 13, 9, 11]. The main reason for this is the fact that it is generally believed to become the standard language for querying XML documents. Furthermore, XQuery is a powerful language in which many typical database queries can be written down very compactly compared to, for example, XSLT. This power however, comes at a price. It is very hard to define the full semantics of XQuery in an elegant and concise manner[3, 2, 4].

From an academic point of view, the need arises for a sublanguage of XQuery, which is nearly as powerful as the full language but has an elegant syntax and semantics that can be written down in just a couple of pages. Similar efforts have been made for XPath[18] and XSLT[7] and have played important roles in practical and theoretical research[10, 16, 15, 14].

The definition of such a language enables us to investigate certain aspects of the XQuery language like

- the power of recursion in XQuery and possible syntactical restrictions that allow us to control this power,
- the complexity of deciding query equivalence for purposes like query optimization,
- the functional character of XQuery, compared to functional programming languages like LISP, ML, et cetera
- the roles of XPath expressions inside XQuery in terms of expressive power and query optimization and
- the relationship between XQuery expressions and the classical well-understood concept of generic database queries.

In this paper we define LiXQuery, a sublanguage of XQuery that is usefull for educational and research purposes and that has several interesting properties, which are easy to prove and can be transposed to the full XQuery language.

The remainder of this paper is organized as follows. In Section 2 we discuss the syntax of LiXQuery and some of the important design choices we made. In Section 3 we give some examples to illustrate the LiXQuery syntax and semantics and Section 4 gives a short introduction to the formal semantics of LiXQuery. Finally, in Section 5 we will point out some interesting research directions that may benefit from this work.

2. SYNTAX AND DESIGN CHOICES

2.1 Syntax

The syntax of LiXQuery is given in Fig. 1 as an abstract syntax, i.e., it assumes that extra brackets and precedence rules are added for disambiguation.

All queries in LiXQuery are syntactically correct in XQuery and their LiXQuery semantics is consistent with those of the XQuery counterparts. Built-in functions for manipulation of basic values are omitted. The non-terminal $\langle Name \rangle$ refers to the set of names $\mathcal{N}$ which we will not describe in detail here except that the names are strings that must start with a letter or "_". The non-terminal $\langle String \rangle$ refers to strings that are enclosed in double quotes such as in `"abc"` and $\langle Integer \rangle$ refers to integers such as `100`, `+100`, and `-100`.[1] Therefore the sets associated with $\langle Name \rangle$, $\langle String \rangle$ and $\langle Integer \rangle$ are pairwise disjoint.

The ambiguity between rule [5] and [24] is resolved by giving precedence to [5], and for path expressions we will assume that the operators "`/`" and "`//`" (rule [18]) are

*Philippe Michiels and Roel Vercammen are supported by IWT – Institute for the Encouragement of Innovation by Science and Technology Flanders, grant numbers 31016 and 31581.

[1]Integers are the only numeric type that exists in LiXQuery.

[1] ⟨*Query*⟩ → (⟨*FunDef*⟩ "`;`")* ⟨*Expr*⟩
[2] ⟨*FunDef*⟩ → "`declare` `function`" ⟨*Name*⟩
"`(`" (⟨*Var*⟩ ("`,`" ⟨*Var*⟩)*)? "`)`"
"`{`" ⟨*Expr*⟩ "`}`"
[3] ⟨*Expr*⟩ → ⟨*Var*⟩ | ⟨*BuiltIn*⟩ | ⟨*IfExpr*⟩ |
⟨*ForExpr*⟩ | ⟨*LetExpr*⟩ | ⟨*Concat*⟩ |
⟨*AndOr*⟩ | ⟨*ValCmp*⟩ | ⟨*NodeCmp*⟩ |
⟨*AddExpr*⟩ | ⟨*MultExpr*⟩ | ⟨*Union*⟩ |
⟨*Step*⟩ | ⟨*Filter*⟩ | ⟨*Path*⟩ |
⟨*Literal*⟩ | ⟨*EmpSeq*⟩ | ⟨*Constr*⟩ |
⟨*TypeSw*⟩ | ⟨*FunCall*⟩
[4] ⟨*Var*⟩ → "`$`" ⟨*Name*⟩
[5] ⟨*BuiltIn*⟩ → "`doc(`" ⟨*Expr*⟩ "`)`" |
"`name(`" ⟨*Expr*⟩ "`)`" |
"`string(`" ⟨*Expr*⟩ "`)`" |
"`xs:integer(`" ⟨*Expr*⟩ "`)`" |
"`root(`" ⟨*Expr*⟩ "`)`" |
"`concat(`" ⟨*Expr*⟩, ⟨*Expr*⟩ "`)`" |
"`true()`" | "`false()`" |
"`not(`" ⟨*Expr*⟩ "`)`" | "`count(`" ⟨*Expr*⟩ "`)`" |
"`position()`" | "`last()`"
[6] ⟨*IfExpr*⟩ → "`if (`" ⟨*Expr*⟩ "`)`"
"`then`" ⟨*Expr*⟩ "`else`" ⟨*Expr*⟩
[7] ⟨*ForExpr*⟩ → "`for`" ⟨*Var*⟩ ("`at`" ⟨*Var*⟩)? "`in`" ⟨*Expr*⟩
"`return`" ⟨*Expr*⟩
[8] ⟨*LetExpr*⟩ → "`let`" ⟨*Var*⟩ "`:=`" ⟨*Expr*⟩
"`return`" ⟨*Expr*⟩
[9] ⟨*Concat*⟩ → ⟨*Expr*⟩ "`,`" ⟨*Expr*⟩
[10] ⟨*AndOr*⟩ → ⟨*Expr*⟩ ("`and`" | "`or`") ⟨*Expr*⟩
[11] ⟨*ValCmp*⟩ → ⟨*Expr*⟩ ("`=`" | "`<`") ⟨*Expr*⟩
[12] ⟨*NodeCmp*⟩ → ⟨*Expr*⟩ ("`is`" | "`<<`") ⟨*Expr*⟩
[13] ⟨*AddExpr*⟩ → ⟨*Expr*⟩ ("`+`" | "`-`") ⟨*Expr*⟩
[14] ⟨*MultExpr*⟩ → ⟨*Expr*⟩ ("`*`" | "`idiv`") ⟨*Expr*⟩
[15] ⟨*Union*⟩ → ⟨*Expr*⟩ "`|`" ⟨*Expr*⟩
[16] ⟨*Step*⟩ → "`.`" | "`..`" | ⟨*Name*⟩ |
"`@`" ⟨*Name*⟩ | "`*`" |
"`@*`" | "`text()`"
[17] ⟨*Filter*⟩ → ⟨*Expr*⟩ "`[`" ⟨*Expr*⟩ "`]`"
[18] ⟨*Path*⟩ → ⟨*Expr*⟩ ("`/`" | "`//`") ⟨*Expr*⟩
[19] ⟨*Literal*⟩ → ⟨*String*⟩ | ⟨*Integer*⟩
[20] ⟨*EmpSeq*⟩ → "`()`"
[21] ⟨*Constr*⟩ → "`element`" "`{`" ⟨*Expr*⟩ "`}`" "`{`" ⟨*Expr*⟩ "`}`" |
"`attribute`" "`{`" ⟨*Expr*⟩ "`}`" "`{`" ⟨*Expr*⟩ "`}`" |
"`text`" "`{`" ⟨*Expr*⟩ "`}`" |
"`document`" "`{`" ⟨*Expr*⟩ "`}`"
[22] ⟨*TypeSw*⟩ → "`typeswitch (`" ⟨*Expr*⟩ "`)`"
("`case`" ⟨*Type*⟩ "`return`" ⟨*Expr*⟩)+
"`default`" "`return`" ⟨*Expr*⟩
[23] ⟨*Type*⟩ → "`xs:boolean`" | "`xs:integer`" |
"`xs:string`" | "`element()`" |
"`attribute()`" | "`text()`" |
"`document-node()`"
[24] ⟨*FunCall*⟩ → ⟨*Name*⟩ "`(`" (⟨*Expr*⟩ ("`,`" ⟨*Expr*⟩)*)? "`)`"

Figure 1: Syntax for LiXQuery queries and expressions

left associative and are preceded by the filter operation (rule [17]) in priority.

2.2 Informal Semantics

Since we assume that the reader is already somewhat familiar with XQuery we only describe here the semantics of some of the less common expressions. More precisely, we will discuss the semantics of rules [5], [11], [12], [15], [21], [22], and [23].

In rule [5] the built-in functions are declared. The function `doc()` returns the document of the XML document with the name that was given as its argument, e.g., `doc("file.xq")` indicates the document root of the content of the XML document `file.xq`. The function `name()` returns the name of an element node or attribute node. The function `string()` gives the string value of an attribute node or text node, and converts integers to strings. The function `xs:integer()`[2] converts strings to integers. The function `root()` gives for a node the root of its tree. The function `concat()` concatenates strings. Rule [11] introduces the comparison operators for basic values. These comparison operators have existential semantics, i.e., they are true for two sequences if there is a basic value in one sequence and a basic value in the other sequence such that the comparison holds between these two basic values. Rule [12] gives the comparison operators for nodes where "`is`" detects the equality of nodes and "`<<`" compares nodes in document order. Rule [15] expresses the union of two node sequences, i.e., it returns a sequence of nodes that contains exactly all the nodes in the operands, contains no duplicates and is sorted in document order. Rule [21] gives the constructors for each type of node. The semantics of "`element` $\{e_1\}\{e_2\}$" is that an element node with name e_1 and content e_2 is created. The semantics of "`attribute` $\{e_1\}\{e_2\}$" is that an attribute node with name e_1 and value e_2 is created. The semantics of "`text` $\{e\}$" is that a text node with value e is created. The semantics of "`document` $\{e\}$" is that a document node with attributes and content as in e is created. Rules [22] and [23] define the typeswitch-expression that checks whether a value belongs to certain types and for the first type that matches returns a certain value.

[2] "`xs:`" indicates a namespace. Although we do not handle namespaces we use them here to be compatible with XQuery.

2.3 Syntactic Sugar

To allow for a shorter notation of certain very common expressions we introduce some short-hands.

2.3.1 The Empty Function

The function `empty()` is assumed to be declared as follows:

```
declare function empty($sequence){
  count($sequence) = 0
};
```

2.3.2 Quantified Formulas

The expression "`some` v `in` e_1 `satisfies` e_2" is introduced as a shorthand for "`not(empty(for` v `in` e_1 `return if (`e_2`) then` v `else ()))`", and "`every` v `in` e_1 `satisfies` e_2" is introduced as a shorthand for "`empty(for` v `in` e_1 `return if (`e_2`) then () else` v`)`".

2.3.3 FLWOR Expression

When `for` and `let` expressions are nested we allow that the intermediate "`return`" is removed. E.g., "`for` v_1

`in` e_1 `return let $`v_2 `:=` e_2 `return` e_3" may be written as "`for $`v_1 `in` e_1 `let $`v_2 `:=` e_2 `return` e_3". Furthermore we allow in `for` and `let` expressions the shorthand "`where` e_1 `return` e_2" for "`return if` e_1 `then` e_2 `else ()`".

2.3.4 Coercion

Let e_1 (or e_2) have the form "`string(`e`)`" where the result of e is a sequence containing a single text node or a single attribute node. Then e_1 (or e_2) can be replaced by e in the following expressions: "`xs:integer(`e_1`)`", "`concat(`e_1`,`e_2`)`", "e_1`=`e_2", "e_1`<`e_2" and "`attribute{`e_1`}{`e_2`}`".

2.4 Design Choices

We have learned from experience that the XQuery standard contains numerous features that are important for designing a practical and efficient query language. However, many of these features do not add any expressive power to the language, while others are not essential for understanding typical queries written in XQuery.

Therefore, we choose to omit a number of standard XQuery features. However, to ensure the validity of LiXQuery, we designed it as a proper sublanguage. Specifically, we made sure that all syntactically valid LiXQuery expressions also satisfy the XQuery syntax. Moreover, the LiXQuery semantics is defined in such a way that the set of possible results of a query evaluated using our semantics will be a proper subset of the result of the set of possible results of the same query evaluated with the full XQuery semantics. Of course, the lack of a complete formal semantics for XQuery does not allow us to prove that relation.

The most visible feature we dropped from XQuery are *types* (and consequently *type coercion*). Types are very useful in XQuery for numerous reasons. But unfortunately, types –especially type coercions– add lots of complexity to a formal semantic definition of a language. And since types are optional in XQuery anyway, we decided to omit them for our sub-language.

Secondly, we removed most of the navigational axes, keeping only the `child`, `parent`, `self` and `descendant-or-self` axes. It has been proven that all other axes can be simulated using the aforementioned ones. Additionally, these extra axes are rarely used in common path expressions.

Finally, we omitted primitive data-types, the `order by` clause, namespaces, comments, programming instructions, entities, and almost all of the built-in functions and operators. For these features we argue that they are necessary to specify a full-fledged query language, yet add too much overhead to incorporate them in a concise, yet formal semantics description.

3. INSTRUCTIVE EXAMPLES

In this section we discuss LiXQuery informally and provide a short example. The query in Fig. 2(a) restructures a list of parts, containing information about their containing parts, to an embedded list of the parts with their subparts [5]. For instance, the document of Fig. 2(b) will be transformed into that of Fig. 2(c). The query starts with the definition of the function `oneLevel`. This is followed by the let-clause that defines the variable `$list` whose value is the `partList` element on the file `partList.xml`. Then a new element is returned with name `intList` and which has as content the result of the function `oneLevel` that is called for each `part`-element `$p` in the `$list` element that has no `partOf`-attribute. The function `oneLevel` constructs a new `part`-element, with one attribute. It is named `partId` and its value is the string of the `partId` attribute of the element `$p` (the second parameter of `oneLevel`). Furthermore the element `part` has a child-element `$s` for each of the parts in the first parameter `$l` and which is part of `$p`. For each such an `$s` the function `oneLevel` is called recursively. If the file `partList.xml` contains Fig. 2(b) the result is shown in Fig. 2(c).

The following example shows how the `ancestor` axis can be simulated in LiXQuery.

```
declare function ancestor($s) {
(: retrieves all anc's of the nodes in $s :)
for $node in $s
  for $anc in root($node)//.
  where some $v in $anc//(*,@*,text())
        satisfies $v is $node
  return $anc
};
```

4. FORMAL SEMANTICS

Due to space limitations, we can only provide a short introduction to the formal semantics of LiXQuery. We refer to [12] for the full description. We will use following notations: the set $\mathcal{A}$ denotes the set of all atomic values, $\mathcal{V}$ is the set of all nodes, $\mathcal{S} \subseteq \mathcal{A}$ is the set of all strings, and $\mathcal{N} \subseteq \mathcal{S}$ is the set of strings that may be used as tag names. The set $\mathcal{V}$ is partitioned into the sets of document nodes ($\mathcal{V}^d$), element nodes ($\mathcal{V}^e$), attribute nodes ($\mathcal{V}^a$), and text nodes ($\mathcal{V}^t$).

4.1 Store

Expressions will be evaluated against an *XML store* which contains XML fragments. This store contains the fragments that are created as intermediate results, but also the web documents that are accessed by the expression. Although in practice these documents are materialized in the store when they are accessed for the first time, we will assume here that all documents are in fact already in the store when the expression is evaluated.

DEFINITION 4.1 (XML STORE). *An* XML store *is a 6-tuple* $St = (V, E, <, \nu, \sigma, \delta)$ *with*

- V *is a finite subset of* $\mathcal{V}$*; we write* V^d *for* $V \cap \mathcal{V}^d$ *(resp.* V^e *for* $V \cap \mathcal{V}^e$*,* V^a *for* $V \cap \mathcal{V}^a$*,* V^t *for* $V \cap \mathcal{V}^t$*);*
- (V, E) *is an acyclic directed graph (with nodes* V *and directed edges* E*) where each node has an in-degree of at most one, and hence it is composed of trees; if* $(m, n) \in E$ *then we say that* n *is a child*

```
declare function oneLevel($l,$p) {
  element { "part" } {
    attribute { "partId" }{ $p/@partId },
    for $s in $l//part
    where $s/@partOf=$p/@partId
    return oneLevel($l,$s)
  }
};

let $list := doc("partList.xml")/partList
return
  element { "intList" } {
    for $p in $list//part[empty(@partOf)]
    return oneLevel($list,$p)
  }
```

(a)

```
<?xml version ="1.0"?>
<partList>
  <part partId="1"/>
  <part partId="2" partOf="1"/>
  <part partId="3" partOf="1"/>
  <part partId="4" partOf="3"/>
  <part partId="5"/>
  <part partId="6" partOf="5"/>
</partList>
```

(b)

```
<intList>
  <part partId="1">
    <part partId="2"/>
    <part partId="3">
      <part partId="4"/>
    </part>
  </part>
  <part partId="5">
    <part partId="6"/>
  </part>
</intList>
```

(c)

Figure 2: A LiXQuery query (a), a document (b) and the result of the query applied to the document (c).

of m;[3] *we denote by E^* the reflexive transitive closure of E;*

- *$<$ is a strict partial order on V that compares exactly the different children of a common node, hence for two distinct nodes n_1 and n_2 it holds that $((n_1 < n_2) \vee (n_2 < n_1)) \Leftrightarrow \exists m \in V((m, n_1) \in E \wedge (m, n_2) \in E)$*
- *$\nu : V^e \cup V^a \rightarrow \mathcal{N}$ labels the element and attribute nodes with their* node name*;*
- *$\sigma : V^a \cup V^t \rightarrow \mathcal{S}$ labels the attribute and text nodes with their* string value*;*
- *$\delta : \mathcal{S} \rightarrow \mathcal{V}^d$ a partial function that associates with an URI or a file name, a document node. It is called the* document function*. This function represents all the URIs of the Web and all the names of the files, together with the documents they contain. We suppose that all these documents are in the store.*

The following properties have to hold for an XML store:

- *each document node of V^d is the root of a tree and has only one child element;*
- *attribute nodes of V^a and text nodes of V^t do not have any children;*
- *in the $<$ -order attribute children precede the element and text children, i.e. if $n_1 < n_2$ and $n_2 \in V^a$ then $n_1 \in V^a$;*
- *there are no adjacent text children, i.e. if $n_1, n_2 \in V^t$ and $n_1 < n_2$ then there is an $n_3 \in V^e$ with $n_1 < n_3 < n_2$;*
- *for all text nodes n_t of V^t holds $\sigma(n_t) \neq$ "";*
- *all the attribute children of a common node have a different name, i.e. if $(m, n_1), (m, n_2) \in E$ and $n_1, n_2 \in V^a$ then $\nu(n_1) \neq \nu(n_2)$.*

Similarly to XQuery there exists a total order over all nodes in the store, called document order. This order is uniquely defined for nodes within the same tree. However, the XQuery Formal Semantics states that each implementation can choose how the root nodes are ordered, as long as the document order is stable during the evaluation of a query.

[3]As opposed to the terminology of XQuery, we consider attribute nodes as children of their associated element node. The definitions of parent, descendant and ancestor are straightforward.

Definition 4.2 (Document Order of a Store). *A* document order $\ll$ *of a store St is a total order on V such that*

1. *if $(n_1, n_2) \in E^*$ and $n_1 \neq n_2$ then $n_1 \ll_{St} n_2$;*
2. *if $(n_1, n_2) \in E^*$ and $n_1 < n_3$ then $(n_2 \ll_{St} n_3)$;*
3. *if $(n_1, n_2), (n_1, n_4) \in E^*$ and $n_2 \ll_{St} n_3 \ll_{St} n_4$ then $(n_1, n_3) \in E^*$.*

1. and 2. define the preorder in a tree. 3. says that the nodes of a tree are clustered.

From this definition follows that we can have more than one document order of a store St, but we choose a fixed document order here that we denote by $\ll_{St}$.

The set of items in a sequence l is denoted as $\mathbf{Set}(l)$. Given a sequence of nodes l in an XML store St we let $\mathbf{Ord}_{St}(l)$ denote the unique sequence $l' = \langle y_1, \ldots, y_m \rangle$ such that $\mathbf{Set}(l) = \mathbf{Set}(l')$ and $y_1 \ll_{St} \ldots \ll_{St} y_m$.

4.2 Environment

Expressions are also evaluated against an environment. The environment contains variable bindings that are currently in scope, function definitions, and the context used for the evaluation of path expressions. Assuming that $\mathcal{X}$ is the set of LiXQuery-expressions this environment is defined as follows.

Definition 4.3 (Environment). *An* environment *of an XML store St is a tuple $En = (\mathbf{a}, \mathbf{b}, \mathbf{v}, \mathbf{x}, \mathbf{k}, \mathbf{m})$ with*

1. *a partial function $\mathbf{a} : \mathcal{N} \rightarrow \mathcal{N}^*$ that maps a function name to its formal arguments; it is used in rule [1,2,24];*

2. *a partial function* $\mathbf{b} : \mathcal{N} \rightarrow \mathcal{X}$ *that maps a function name to the body of the function; it is also used in rules [1,2,24];*
3. *a partial function* $\mathbf{v} : \mathcal{N} \rightarrow (\mathcal{V} \cup \mathcal{A})^*$ *that maps variable names to their values;*
4. $\mathbf{x}$ *which is undefined or an item of St and indicates the context item – it is used in rule [16,17,18];*
5. $\mathbf{k}$ *which is undefined or an integer and gives the position of the context item in the context sequence; it is used in rule [5,17,18];*
6. $\mathbf{m}$ *which is undefined or an integer and gives the size of the context sequence; it is used in rule [5, 17,18].*

If En is an environment, n a name and y an item then we let $En[\mathbf{a}(n) \mapsto y]$ ($En[\mathbf{b}(n) \mapsto y]$, $En[\mathbf{v}(n) \mapsto y]$) denote the environment that is equal to En except that the function $\mathbf{a}$ ($\mathbf{b}$, $\mathbf{v}$) maps n to y. Similarly, we let $En[\mathbf{x} \mapsto y]$ ($En[\mathbf{k} \mapsto y]$, $En[\mathbf{m} \mapsto y]$) denote the environment that is equal to En except that $\mathbf{x}$ ($\mathbf{k}$, $\mathbf{n}$) is defined as y if $y \neq \perp$ and undefined otherwise.

We write $St, En \vdash e \Rightarrow (St', v)$ to denote that the evaluation of expression e against the XML store St and environment En of St may result in the new XML store St' and value v of St'.

4.3 Semantic Rules

In what follows we discuss some of the reasoning rules that are used to define the semantics of LiXQuery. For the full set of rules, we refer to [12]. Each rule consists of a set of premises and a conclusion of the form $St, En \vdash e \Rightarrow (St', v)$. The free variables in the rules are always assumed to be universally quantified.

For-expression (Rule [7]) The rule for

$$\texttt{for } \$s \texttt{ at } \$s' \texttt{ in } e \texttt{ return } e'$$

specifies that first e is evaluated and then e' for each item in the result of e but with $\$s$ and $\$s'$ in the environment bound to the respectively the item in question and its position in the result of e. Finally the results for each item are concatenated to a single sequence.

$$\frac{\begin{array}{c} St, En \vdash e \Rightarrow (St_0, \langle x_1, \ldots, x_m \rangle) \\ St_0, En[\mathbf{v}(\$s) \mapsto x_1][\mathbf{v}(\$s') \mapsto 1] \vdash e' \Rightarrow (St_1, v_1) \\ \ldots \\ St_{m-1}, En[\mathbf{v}(\$s) \mapsto x_m][\mathbf{v}(\$s') \mapsto m] \vdash e' \Rightarrow (St_m, v_m) \end{array}}{St, En \vdash \texttt{for } \$s \texttt{ at } \$s' \texttt{ in } e \texttt{ return } e' \Rightarrow (St_m, v_1 \circ \ldots \circ v_m)}$$

Constructors (Rule [21]) Constructors are the only operations that create a new store. More precisely, the inference rules for constructors are the only rules that have a result store in the conclusion that is not the input store or a result store of one of the subexpressions.

Before we proceed with the presentation of the rule for the element constructor, we first introduce the notion of *deep equality*. This defines what it means for two nodes in an XML store to represent the same XML fragment.

DEFINITION 4.4 (DEEP EQUAL). *Given the XML store* $St = (V, E, <, \nu, \sigma, \delta)$ *and two nodes* n_1 *and* n_2 *in* St. n_1 *and* n_2 *are said to be* deep equal, *denoted as* $\mathbf{DpEq}_{St}(n_1, n_2)$, *if* n_1 *and* n_2 *refer to two isomorphic trees, i.e., there is a one-to-one function* $h : C_{n_1} \rightarrow C_{n_2}$ *with* $C_{n_i} = \{n | (n_i, n) \in E^*\}$ *for* $i = 1, 2$, *such that for each* $n, n' \in C_{n_1}$ *it holds that (1) if* $n \in \mathcal{V}^d$ ($\mathcal{V}^e$, $\mathcal{V}^a$, $\mathcal{V}^t$) *then* $h(n) \in \mathcal{V}^d$ ($\mathcal{V}^e$, $\mathcal{V}^a$, $\mathcal{V}^t$), *(2) if* $\nu(n) = s$ *then* $\nu(h(n)) = s$, *(3) if* $\sigma(n) = s'$ *then* $\sigma(h(n)) = s'$, *(4)* $(n, n') \in E$ *iff* $(h(n), h(n')) \in E$ *and (5) if* $n, n' \notin \mathcal{V}^a$ *then* $n < n'$ *iff* $h(n) < h(n')$.

The semantics of the element constructor

$$\texttt{element}\{e'\}\{e''\}$$

is defined as follows. First e' is evaluated and assumed to result in a single legal element name. The e'' is evaluated and for the result we create a new store St_3 that contains the new element with the result of e' as its name and with contents that are deep-equivalent with the result of e'' if we compare them item by item. Finally we add St_3 to the original store and return the newly created element node.

$$\frac{\begin{array}{c} St, En \vdash e' \Rightarrow (St_1, \langle s \rangle) \\ s \in \mathcal{N} \qquad St_1, En \vdash e'' \Rightarrow (St_2, \langle n_1, \ldots, n_m \rangle) \\ n_1, \ldots, n_m \in \mathcal{V} \qquad St_4 = St_2 \cup St_3 \\ n \in V_{St_3} \Rightarrow (r, n) \in E^*_{St_3} \qquad r \in \mathcal{V}^e \qquad \nu_{St_3}(r) = s \\ \mathbf{Ord}_{St_3}(\{n' | (r, n') \in E_{St_3}\}) = \langle n'_1, \ldots, n'_m \rangle \\ \mathbf{DpEq}_{St_4}(n_1, n'_1) \qquad \ldots \qquad \mathbf{DpEq}_{St_4}(n_m, n'_m) \\ \forall\, n, n' \in \mathcal{V}((n \ll_{St_2} n') \Rightarrow (n \ll_{St_4} n')) \end{array}}{St, En \vdash \texttt{element}\{e'\}\{e''\} \Rightarrow (St_4, \langle r \rangle)}$$

Similar rules exist for attribute and text node construction. Note that the constructor rules are the only nondeterministic rules in LiXQuery, since there are several possible document orders for the new store St_4, which are only partially restricted by earlier choices of a document order St_2.

5. RESEARCH OPPORTUNITIES

The formal specification of LiXQuery gives us the opportunity to study some aspects of XQuery more formally. In this section we will discuss a few possible research directions in which we can use the LiXQuery language as a formal foundation.

A first application of LiXQuery is that of formally specifying extensions to, or alternative subsets of XQuery. For example, the language XQBE[8] (XQuery By Example) is a powerful graphical query language for XML with a well-defined formal semantics. The implementation of XQBE translates visual queries to expressions within a subset of XQuery. Translating XQBE to LiXQuery expressions instead of XQuery expressions would facilitate the process of formally proving the correspondence between the semantics of XQBE and generated XQuery expressions. A more fundamental extension to XQuery that can be formally described using LiXQuery

are updates in XQuery. In [17] an extension to XQuery to incorporate update operations is defined, but this work lacks a formal semantics, hence it is for example unclear whether node identity is preserved when moving or replacing a node.

Another possible research topic that may benefit from our LiXQuery definition is XQuery optimization, where we could use our formal semantics to prove that we can replace some expressions in a query by equivalent expressions, which are less expensive to evaluate.

Finally, we can use LiXQuery for studying the expressive power of some typical XQuery constructs, since LiXQuery provides us a compact and formal foundation needed to perform this study. Fragments of LiXQuery can be defined and studied to identify a number of structural properties, similar to what was done for XPath fragments by Benedikt, Fan, and Kuper in [6]. These fragments can also be used for studying the complexity of query evaluation in XQuery fragments, comparable to the study of the complexity of XPath query evaluation performed by Gottlob, Koch, and Pichler [10].

6. DISCUSSION

In this work we have introduced a fragment of XQuery called LiXQuery along with a formal and concise description of its semantics that is consistent with the formal semantics of XQuery. We claim that this fragment captures the essence of XQuery as a query language and can therefore be used for educational purposes, e.g., teaching XQuery, and research purposes, e.g., investigating the expressive power of XQuery fragments.

7. REFERENCES

[1] XML query (XQuery). `http://www.w3.org/XML/Query`.

[2] XQuery 1.0 and XPath 2.0 data model, W3C working draft 29 october 2004. `http://www.w3.org/TR/2003/WD-xpath-datamodel/`.

[3] XQuery 1.0 and XPath 2.0 formal semantics, W3C working draft 20 february 2004. `http://www.w3.org/TR/2004/WD-xquery-semantics/`.

[4] XQuery 1.0 and XPath 2.0 functions and operators, W3C working draft 29 october 2004. `http://www.w3.org/TR/2003/WD-xpath-functions/`.

[5] XML query use cases, 1.8.4.1. 2003. `http://www.w3.org/TR/xquery-use-cases/`.

[6] M. Benedikt, W. Fan, and G. M. Kuper. Structural properties of XPath fragments. In *International Conference on Database Theory*, 2003.

[7] G. J. Bex, S. Maneth, and F. Neven. A formal model for an expressive fragment of XSLT. *Information Systems*, 27:21–39, 2002.

[8] A. Campi, D. Braga, and S. Ceri. XQBE (XQuery by example): a visual interface to the standard XML query language. *ACM Transactions on Database Systems, June 2005*, 2005.

[9] D. Chamberlin. XQuery: An XML query language, tutorial overview. *IBM Systems Journal*, 41(4), 2002.

[10] G. Gottlob, C. Koch, and R. Pichler. The complexity of XPath query evaluation. In *Proc. of the 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Priciples of Database Systems (PODS)*, San Diego (CA), 2003.

[11] J. Harbarth. XQuery 1.0 primer. 2003. `http://www.softwareag.com/xml/tools/xquery_primer.pdf`.

[12] J. Hidders, J. Paredaens, R. Vercammen, and S. Demeyer. A light but formal introduction to XQuery. In *Proceedings of the Second International XML Database Symposium (XSym 2004)*, number 2186 in LNCS, pages 5–20, Toronto, Canada, 2004. Springer. `http://www.adrem.ua.ac.be/pub/lixquery.pdf`.

[13] H. Katz, D. Chamberlin, D. Draper, M. Fernández, M. Kay, J. Robie, M. Rys, J. Siméon, J. Tivy, and P. Wadler, editors. *XQuery from the Experts: A Guide to the W3C XML Query Language*. Addison-Wesley, 2004.

[14] S. Kepser. A simple proof of the turing-completeness of XSLT and XQuery. In T. Usdin, editor, *Extreme Markup Languages 2004*, 2004.

[15] W. Martens and F. Neven. Frontiers of tractability for typechecking simple XML transformations. In *Proc. of the 23rd ACM SIGACT-SIGMOD-SIGART Symposium on Priciples of Database Systems (PODS)*, Paris, France, 2004.

[16] M. Marx. XCPath, the first order complete XPath dialect. In *Proc. of the 23rd ACM SIGACT-SIGMOD-SIGART Symposium on Priciples of Database Systems (PODS)*, Paris, France, 2004.

[17] I. Tatarinov, Z. G. Ives, A. Y. Halevy, and D. S. Weld. Updating XML. In *Proc. of the ACM SIGMOD 2001 International Conference on Management of Data*, 2001.

[18] P. Wadler. Two semantics for XPath, 1999. `http://www.cs.bell-labs.com/who/wadler/topics/xml.html`.

From Databases to Dataspaces: A New Abstraction for Information Management

Michael Franklin
University of California, Berkeley

Alon Halevy
Google Inc. and U. Washington

David Maier
Portland State University

Abstract

The development of relational database management systems served to focus the data management community for decades, with spectacular results. In recent years, however, the rapidly-expanding demands of "data everywhere" have led to a field comprised of interesting and productive efforts, but without a central focus or coordinated agenda. The most acute information management challenges today stem from organizations (e.g., enterprises, government agencies, libraries, "smart" homes) relying on a large number of diverse, interrelated data sources, but having no way to manage their *dataspaces* in a convenient, integrated, or principled fashion. This paper proposes dataspaces and their support systems as a new agenda for data management. This agenda encompasses much of the work going on in data management today, while posing additional research objectives.

1. Introduction

A Database Management System (DBMS) is a generic repository for the storage and querying of structured data. A DBMS offers a suite of interrelated services and guarantees that enables developers to focus on the specific challenges of their applications, rather than on the recurring challenges involved in managing and accessing large amounts of data consistently and efficiently.

Unfortunately, in data management scenarios today it is rarely the case that all the data can be fit nicely into a conventional relational DBMS, or into any other single data model or system. Instead, developers are more often faced with a set of loosely connected data sources and thus must individually and repeatedly address low-level data management challenges across heterogeneous collections. These challenges include: providing search and query capability; enforcing rules, integrity constraints, naming conventions, etc.; tracking lineage; providing availability, recovery, and access control; and managing evolution of data and metadata.

Such challenges are ubiquitous – they arise in enterprises (large or small): within and across government agencies, large science-related collaborations, libraries (digital or otherwise), battlefields, in "smart" homes, and even on one's PC desktop or other personal devices. In each of these scenarios, however, there is some identifiable scope and control across the data and underlying systems, and hence one can identify a space of data, which, if managed in a principled way, will offer significant benefits to the organization.

In this article we introduce dataspaces as a new abstraction for data management in such scenarios and we propose the design and development of DataSpace Support Platforms (DSSPs) as a key agenda item for the data management field. In a nutshell, a DSSP offers a suite of interrelated services and guarantees that enables developers to focus on the specific challenges of their applications, rather than on the recurring challenges involved in dealing consistently and efficiently with large amounts of interrelated but disparately managed data. We begin our discussion of dataspaces and DSSPs by placing them in the context of existing systems.

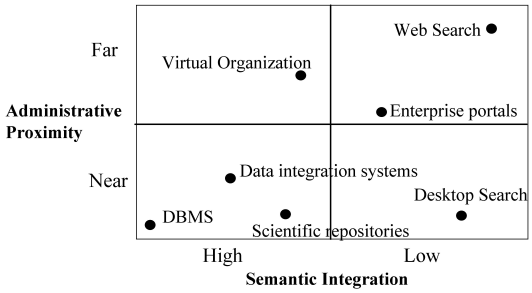


Figure 1. A space of data management solutions.

1.1 Data Management Architectures

Figure 1 shows a categorization of existing data management solutions along two dimensions. "Administrative Proximity" indicates how *close* the various data sources are in terms of administrative control. "Near" means that the sources are under the same or at least coordinated control, while "Far" indicates a looser coordination tending towards none at all. The closer the administrative control of a group of data sources, the stronger the guarantees (e.g., of consistency, permanence) that can be provided by the data management system.

"Semantic Integration" is a measure of how closely the schemas of the various data sources have been matched. In other words, how well the types, names, units, meanings, etc. of the data in the sources are matched up. At the high end of the spectrum, all data conforms to a single agreed-upon schema. At the low end, there is no schema information at all. In between lay various data integration solutions and approaches based on semi-structured data and controlled vocabularies. This dimension indicates the degree to which semantically rich query processing and data manipulation can be provided across a group of data sources, with higher degrees of integration providing richer functionality.

As shown in the figure, traditional DBMSs represent only one (albeit, an important one) point solution in today's data management environment. DBMSs require all data to be under the control of a single administrative domain and to conform to a single schema. In return for these limitations, a DBMS is able to provide rich data manipulation and query processing with well-understood, strong semantics, as well as strict transactional guarantees for updates, concurrency, and persistence (the so-called "ACID" properties).

An important point in Figure 1 is "data integration systems". In fact, traditionally, data integration and data exchange systems have

aimed to offer many of the purported services of dataspace systems. The distinction is that data integration systems require *semantic integration* before any services can be provided. Hence, although there is not a single schema to which all the data conforms, the system knows the precise relationships between the terms used in each schema. As a result, significant upfront effort is required in order to set up a data integration system.

Dataspaces are not a data integration approach; rather, they are more of a *data co-existence* approach. The goal of dataspace support is to provide base functionality over all data sources, regardless of how integrated they are. For example, a DSSP can provide keyword search over all of its data sources, similar to that provided by existing desktop search systems. When more sophisticated operations are required, such as relational-style queries, data mining, or monitoring over certain sources, then additional effort can be applied to more closely integrate those sources in an incremental, "pay-as-you-go" fashion.

Similar flexibility exists along the administrative proximity dimension of Figure 1. If administrative autonomy is desired then the DSSP will not be able to provide certain guarantees in terms of consistency, durability of updates, etc. As stronger guarantees are desired, more effort can be put into making agreements among the various owners of data sources and opening up certain interfaces (e.g., for commit protocols).

To summarize, the distinguishing properties of dataspace systems are the following:

- A DSSP must deal with data and applications in a wide variety of formats accessible through many systems with different interfaces. A DSSP is required to support *all* the data in the dataspace rather than leaving some out, as with DBMSs.
- Although a DSSP offers an integrated means of searching, querying, updating, and administering the dataspace, often the same data may also be accessible and modifiable through an interface native to the system hosting the data. Thus, unlike a DBMS, a DSSP is not in full control of its data.
- Queries to a DSSP may offer varying levels of service, and in some cases may return *best-effort* or approximate answers. For example, when individual data sources are unavailable, a DSSP may be capable of producing the best results it can, using the data accessible to it at the time of the query.
- A DSSP must offer the tools to create tighter integration of data in the space as necessary.

1.2 A Dataspace Agenda

By all measures, the data management research community remains active, vibrant, and growing. The concern has been raised, however, that the community currently lacks a central focus — a "relational DBMS" equivalent for the new world of disparate decentralized data.[1]. Furthermore, there is a growing feeling among many, that the term "database research" is too restrictive for the breadth of topics being addressed by the community. While it may be possible that the field has simply grown too large to accommodate a single, succinct vision, this paper is intended as one proposal that could help further a discussion of the issues.

The database community has long had a process of self-assessment in which senior researchers meet periodically to survey the state of the field and to identify promising research areas for the future (the most recent of these are the 1998 Asilomar Report [BBC+98] and the 2005 Lowell Self-Assessment [AAB+05]). This paper builds on many of the goals and challenge problems identified in those earlier reports. In fact, much of the research in the data management community already falls squarely into the requirements of dataspaces and DSSPs, including areas such as schema mapping, data integration and model management, uniform search over multiple types of data; combining structured, semi-structured, and unstructured data, approximate query processing; managing and querying uncertain data and its lineage, and stream and sensor data management and processing. Thus, dataspaces can be viewed simply as an umbrella for these varied efforts. As we discuss later, however, we also believe that the holistic view taken by dataspaces and DSSPs can itself lead to a new set of research challenges.

The remainder of this paper is as follows. Section 2 motivates the need for dataspace systems with two prototypical examples. Section 3 describes the logical components of a dataspace and a first attempt at an architecture of a DSSP. Section 4 outlines several research challenges critical to building DSSP, and Section 5 discusses a few perspectives on the agenda. Section 6 concludes.

2. Examples

We begin by describing two dataspace scenarios.

Personal Information Management: The goal of Personal Information Management (PIM) is to offer easy access and manipulation of all of the information on a person's desktop, with possible extension to mobile devices, personal information on the Web, or even all the information accessed during a person's lifetime.

Recent desktop search tools are an important first step for PIM, but are limited to keyword queries. Our desktops typically contain some structured data (e.g., spreadsheets) and there are important associations between disparate items on the desktop. Hence, the next step for PIM is to allow the user to search the desktop in more meaningful ways. For example, "find the list of students who took my database course last quarter", or "compute the aggregate balance of my bank accounts". We would also like to search by association, e.g., "find the email that John sent me the day I came back from Hawaii", or "retrieve the experiment files associated with my SIGMOD paper this year". Finally, we would like to query *about* sources, e.g., "find all the papers where I acknowledged a particular grant", "find all the experiments run by a particular student", or "find all spreadsheets that have a variance column".

The principles of dataspaces in play in this example are that (1) a PIM tool must enable accessing *all* the information on the desktop, and not just an explicitly chosen subset, and (2) while PIM often involves integrating data from multiple sources, we cannot assume users will invest the time to integrate. Instead, most of the time the system will have to provide best-effort results, and tighter integrations will be created only in cases where the benefits will clearly outweigh the investment.

Scientific data management: Consider a scientific research group working on environmental observation and forecasting. They may be monitoring a coastal ecosystem through weather stations, shore- and buoy-mounted sensors and remote imagery. In addition they can be running atmospheric and fluid-dynamics models that simulate past, current and near-future conditions. The computations may require importing data and model outputs from other groups, such as river flows and ocean circulation forecasts. The observations and simulations are the inputs to programs that generate a wide range of data products, for use within the group and by others: comparison plots between observed and simulated data, images of surface-temperature distributions, animations of salt-water intrusion into an estuary.

Such a group can easily amass millions of data products in just a few years. While it may be that for each file, someone in the group knows where it is and what it means, no one person

[1] For instance, this issue was raised and discussed perhaps most publicly at the CIDR 2005 Conference

may know the entire holdings nor what every file means. People accessing this data, particularly from outside the group, would like to search a master inventory that had basic file attributes, such as time period covered, geographic region, height or depth, physical variable (salinity, temperature, wind speed), kind of data product (graph, isoline plot, animation), forecast or hindcast, and so forth. Once data products of interest are located, understanding the lineage is paramount in being able to analyze and compare products: What code version was used? Which finite element grid? How long was the simulation time step? Which atmospheric dataset was used as input?

Soon, such groups will need to federate with other groups to create scientific dataspaces of regional or national scope. They will need to easily export their data in standard scientific formats, and at granularities (sub-file or multiple file) that don't necessarily correspond to the partitions they use to store the data. Users of the federated dataspace may want to see collections of data that cut across the groups in the federation, such as all observations and data products related to water velocity, or all data related to a certain stretch of coastline for the past two months. Such collections may require local copies or additional indices for fast search.

This scenario illustrates several dataspace requirements, including (1) a dataspace-wide catalog, (2) support for data lineage and (3) creating collections and indexes beyond what any one participating source supplies.

3. Dataspaces

We now describe the logical components of a dataspace and the services we expect from a DSSP.

3.1 Logical Components of Dataspaces

A dataspace (see Figure 2) should contain all of the information relevant to a particular organization regardless of its format and location, and model a rich collection of relationships between data repositories. Hence, we model a dataspace as a set of *participants* and *relationships*.

The participants in a dataspace are the individual data sources: they can be relational databases, XML repositories, text databases, web services and software packages. They can be stored or streamed (managed locally by data stream systems), or even sensor deployments.

Some participants may support expressive query languages, while others are opaque and offer only limited interfaces for posing queries (e.g., structured files, web services, or other software packages). Participants vary from being very structured (e.g., relational databases) to semi-structured (XML, code collections) to completely unstructured. Some sources will support traditional updates, while others may be append-only (for archiving purposes), and still others may be immutable.

A dataspace should be able to model any kind of relationship between two (or more) participants. On the more traditional end, we should be able to model that one participant is a view or a replica of another, or to specify a schema mapping between two participants. We would, however, like to model a much broader set of relationships such as, that source A was manually curated from sources B and C, or that sources E and F were created independently, but reflect the same physical system (e.g., mouse DNA). Relationships may be even less specific, such as that two datasets came from the same source at the same time.

Dataspaces can be nested within each other (e.g., the dataspace of the CS department is nested within the dataspace of the university), and they may overlap (e.g., the dataspace of the CS department may share some participants with the EE department). Hence, a dataspace must include access rules between disparate dataspaces. In general, there will be cases where the boundaries of a dataspace may be fluid, but we expect that in most of the cases the boundaries will be natural to define.

3.2 Dataspace Services

Along with content heterogeneity comes the need to support multiple styles of access to the content. We envision that DSSPs will allow many different modes of interaction and we aspire to be as general as possible in allowing the application of different services to different types of content.

One of the most basic dataspace services is cataloging data elements from the participants. A catalog is an inventory of data resources, with the most basic information about each, such as source, name, location in source, size, creation date and owner, and so forth. The catalog is infrastructure for most of the other dataspace services, but can also support a basic browse interface across the dataspace for users.

Two of the main services that a DSSP will support are search and query. While DBMSs have excelled at providing support for querying, search has emerged as a primary mechanism for end users to deal with large collections of unfamiliar data. Search has the property that it is more forgiving than query, being based on similarity and providing ranked results to end users, and supporting interactive refinement so that users can explore a data set and incrementally improve their results. A DSSP should enable a user to specify a search query and iteratively refine it, when appropriate, to a database-style query. A key tenet of the dataspaces approach is that search should be applicable to all of the contents of a dataspace, regardless of their formats.

Universal search and query should extend to meta-data as well as data. Users should be able to discover relevant data sources and inquire about their completeness, correctness and freshness. In fact, a DSSP should also be aware of *gaps* in its coverage of the domain.

A DSSP will also support updating data. Obviously, the effects of updates will be determined by the mutability of the relevant data sources. A major research issue in dataspaces is the development and provision of guaranteed update semantics in a heterogeneous, highly-autonomous environment.

Other key DSSP services include monitoring, event detection, and support for complex workflows. For example, we may want to set up a computation to happen when a new piece of data arrives, and have the results of that computation distributed to a set of recipient data sources. Similarly, a DSSP should support various forms of data mining and analysis.

Not every participant in a dataspace will necessarily provide the interfaces necessary to support all DSSP functions. Thus, there will be the need to extend data sources in various ways. A source might not actually store its own metadata, so we may require an independent metadata repository for such sources. Information may need to be "externalized" from a source or its context. For example, a list of emergency services agencies from Washington might need to be explicitly labeled "Washington" in order to combine it with similar lists from Oregon and California. Or a scientific dataset might need a superimposed schema. The data elements in a source might be enhanced with annotations, ratings, links to elements in other sources. Monitoring support may need to be provided for participants that lack their own notification service.

3.3 Dataspace Systems

We now outline one possible set of components and architecture for a dataspace system. As depicted in Figure 2, a DSSP offers several interrelated services on the dataspace, some of which are generalizations of components provided by a traditional DBMS.

It is important to keep in mind that unlike a DBMS, a DSSP does not assume complete control over the data in the dataspace. Instead, a DSSP allows the data to be managed by the participant

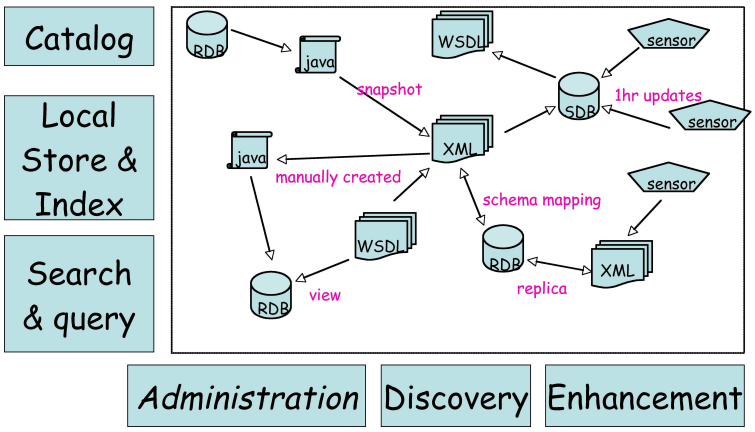


Figure 2. An example dataspace and the components of a dataspace system.

systems, but provides a new set of services over the aggregate of these systems, while remaining sensitive to the autonomy needs of the systems. Furthermore, we may have several DSSPs serving the same dataspace – in a sense, a DSSP can be a personal view on a particular dataspace.

Catalog and Browse: The catalog contains information about all the participants in the dataspace and the relationships among them. The catalog must be able to accommodate a large variety of sources and support differing levels of information about their structure and capabilities. In particular, for each participant, the catalog should include the schema of the source, statistics, rates of change, accuracy, completeness, query answering capabilities, ownership, and access and privacy policies. Relationships may be stored as query transformations, dependency graphs, or sometimes even textual descriptions.

Wherever possible, the catalog should contain a basic inventory of the data elements at each participant: identifier, type, creation date and so forth. It can then support a basic browse capability over the combined inventory of all participants. While not a very scalable interface, it can at least be used to answer questions about the presence or absence of a data element, or determine which participants hold documents of a particular type. Simple scripts run over the participants can extend the capabilities of this interface. For example, computing and storing an MD5 hash of all data elements can help identify duplicated holdings between participants.

On top of the catalog, the DSSP should support a model-management environment that allows creating new relationships and manipulate existing ones (e.g., mapping composition and inversion, merging of schemas and creating unified views of multiple sources).

Search and Query: The component should offer the following capabilities:

(1) Query everything: Users should be able to query any data item regardless of its format or data model. Initially, the DSSP should support keyword queries on any participant. As we gain more information about a participant, we should be able to gradually support more sophisticated queries. The system should support graceful transition between keyword querying, browsing and structured querying. In particular, when answers are given to a keyword (or structured) query, additional query interfaces should be proposed that enable the user to refine the query.

(2) Structured query: Database-like queries should be supported on common interfaces (i.e., mediated schemas) that provide access to multiple sources, or can be posed on a specific data source (using its own schema) with the intention that answers will also be obtained from other sources (as in peer-data management systems). Queries can be posed in a variety of languages (and underlying data models) and should be reformulated into other data models and schemas as best possible, leveraging exact and approximate semantic mappings.

(3) Meta-data queries: The system should support a wide spectrum of meta-data queries. These include (a) including the source of an answer or how it was derived or computed, (b) providing timestamps on the data items that participated in the computation of an answer, (c) specifying which other data items in the dataspace may depend on a particular data item and being able to support hypothetical queries (i.e., *What would change if I removed data item X?*), and (d) querying the sources and degree of uncertainty about the answers.

A DSSP should also support queries locating data, where the answers are data sources rather than specific data items. For example, the system should be able to answer a query such as: *Where can I find data about IBM?*, or *What sources have a* salary *attribute?* Similarly, given an XML document, one should be able to query for XML documents with similar structures, and XML transformations that involve them. Finally, given a fragment of a schema or a web-service description, it should be possible to find similar ones in the dataspace.

(4) Monitoring: All of the above Search and Query services should also be supported in an incremental form that can be applied in real-time to streaming or modified data sources. Monitoring can be done either as a stateless process, in which data items are considered individually, or as a stateful process, where multiple data items are considered. For example, message filtering is a stateless process, whereas windowed aggregate computation is stateful. Complex event detection and alerting are additional functionalities that can be provided as part of an incremental monitoring service.

Local store and index: A DSSP will have a storage and indexing component for the following goals: (1) to create efficiently queryable associations between data objects in different participants, (2) to improve accesses to data sources that have limited access patterns, (3) to enable answering certain queries without accessing the actual data source, and (4) to support high availability and recovery.

The index needs to be highly adaptive to heterogeneous environments. It should take as input any token appearing in the dataspace and return the locations at which the token appears and the roles of each occurrence (e.g., a string in a text file, element in file path, a

value in a database, element in a schema or tag in XML file). Two important aspects of the index are that (1) it identifies information *across* participants when certain tokens appear in multiple ones (in a sense, a generalization of join index). Typically, we may want to build special indexes for this purpose for a certain set of tokens, and (2) it is robust in the face of multiple references to real-world objects, e.g., different ways to refer to a company or person.

We may want to cache certain dataspace fragments (vertical or horizontal) for several purposes including: (1) to build additional indexes on them for supporting more efficient access, (2) to increase availability of data that is stored in participants that may not be reliable, and (3) to reduce the query load on participants that cannot allow ad-hoc external queries.

The Discovery Component: The goal of this component is to locate participants in a dataspace, create relationships between them, and help administrators to refine and tighten these relationships.

Location of participants can take several forms, such as starting a traversal from the root of a directory structure, or trying to locate all the databases on an enterprise network. The component should perform an initial classification according to participant type and content.

Once the participants are discovered, the system should provide an environment for semi-automatically creating relationships and improving and maintaining existing relationships between participants. This involves both finding which pairs of participants are likely to be related to each other, and then proposing relationships (e.g., schema mappings, replicas, containment relationships) that are then verified and refined by a human. Finally, it is important that the discovery component monitor the contents of the dataspace to propose additional relationships over time.

The Source Extension Component: Certain participants may lack significant data management functions. A participant might be no more than a departmental document repository, perhaps with no service other than weekly backups. A DSSP should be able to imbue such a participant with additional capabilities, such as a schema, a catalog, keyword search and update monitoring. Note that it may be necessary to provide these extensions "in-situ", as there can be existing applications or workflows that assume the current formats or directory structures.

This component also supports "value-added" information held by the DSSP, but not present in of the initial participants. Such information can include "lexical crosswalks" between vocabularies, translation tables for coded values, classifications and ratings of documents, and annotations or links attached dataset or document contents. Such information must be able to span participants. For example, in the desktop database a significant amount of effort is put into building associations between items in different applications (e.g., storing connections between presentations, papers and programs that all relate to the same project).

While we imagine that a "full service" DSSP contains all these components, we point out that many of them could be used on their own, to achieve certain price-benefit tradeoffs. For example, a large university initially may only be able to afford a Catalog and Browse service for the campus-wide dataspace, but that could be an improvement over the existing opaqueness of resources. Later, keyword query capabilities might be added, campus-wide or in selected sub-dataspaces. It is important that DSSPs can yield incremental payoff for incremental investment, and not exist only as monolithic solutions. Finally, though we do not describe these in detail, we expect a DSSP to have an administration component and some module that supports "soft" recovery.

4. Research Challenges

This section identifies some of the new challenges that arise in building DSSPs.

4.1 Data models and querying in DSSPs

Data modeling and basic querying: Unlike a DBMS, a DSSP needs to support multiple data models at its core so it accommodates as many types of participants as possible in a natural way.

The data models supported by a DSSP will fall into a hierarchy of expressive power. Every participant in the dataspace supports some data model and some query language appropriate for that model. For example, at the very top (most general) level of the hierarchy are collections of named resources, possibly with basic properties, such as size, creation date and type (e.g., JPEG image, MySQL database). "Query" against this data model corresponds to what a file system typically supports for its directories: name match, find in date range, sort by file size, and so forth. Below the top level, a DSSP should support the bag-of-words data model, implying that we should be able to pose keyword queries on any participant in the dataspace, and hence gain some visibility into the participants in a dataspace.

The semi-structured labeled-graph data model can come one level below the bag-of-words model in the hierarchy. Whenever a participant supports some structure, we should be able to pose simple path or containment queries, or possibly more complex queries based on the semi-structured data model. The goal should be that whenever there is a way of naturally interpreting a path query on a participant, the query processor should attempt to follow such an interpretation.

There will be other data models in the hierarchy, including the relational model, XML with schema, RDF, OWL (the Web Ontology Language). Given an environment, a key challenge is to find methods for interpreting queries in various languages on participants that support certain models. Specifically, how do we reformulate a query posed in a complex language on a source that supports a weaker data model, and conversely, how do we reformulate a query in a simple language on a source that supports a more expressive model and query language (e.g., keyword query on a relational database).

A broader view of querying: To adequately address the needs of dataspace application scenarios and users, a DSSP needs to support a broader approach to querying. Due mostly to the WWW and the ensuing revolution in how people can access information, people have recognized search to be a first-class activity. Computer users realize that a significant portion of their computer-aided activities can be divided into two parts: searching for relevant information, and acting on the found information. Search can come in many flavors, some reminiscent of database querying, such as finding flights for a trip, checking bank balances online, and others closer to keyword search, such as finding appropriate documents within an enterprise and looking for waffle recipes.

Hence, offering intuitive search and query on everything is a key challenge. In fact, from a user's perspective, the distinction between search and query should disappear. Users should start searching in the simplest way and then be directed as appropriate to more specialized search and query interfaces. The system should provide useful suggestions to the user as to what other searches or topics may be of interest given the query. Intuitive visualizations of results also need to be developed to guide users in the right directions.

4.2 Dataspace discovery

A crucial component to building a dataspace is to discover its participants and the relationships between them. A very common problem in today's large enterprises is that they don't even know which

data sources they have throughout the organization. The ultimate goal of dataspace discovery is locate participants in the dataspace, create the relationships between them, and improve the fidelity of the existing relationships between participants. The main components of a dataspace discovery system are (1) locating the participants in the organization, (2) a semi-automatic tool for clustering and finding relationships between participants, and (3) a tool for creating more precise relationships between participants (at the extreme, these are schema mappings).

4.3 Reusing human attention

One of the key properties of dataspaces is that semantic integration evolves over time and only where needed. The most scarce resource available for semantic integration is human attention. Hence, it is crucial that DSSPs know how to reuse human attention, generalize from it, and reuse it for other tasks. The community has already developed methods for reusing human work in creating semantic mappings between data sources, but this capability is only a first step. Other examples of human work that can be reused include annotations (e.g., someone manually relating data items from two different sources), temporary collections of data that are created for a particular task (known as digital workspaces), queries written on the data (which imply certain relationships that may not be known otherwise), and operations on the data (e.g., cutting and pasting values from one column in a spreadsheet into a column in a different spreadsheet). The goal is that previous work should be recorded in the system and leveraged when we try to create additional relationships between participants in the dataspace or try to answer queries over it. We expect Machine Learning techniques to be useful here.

4.4 Dataspace storage and indexing

The key challenges involved in building the local store and indexing component of a DSSP have to do with the heterogeneity of the index. The index should uniformly index all possible data items, whether they are words appearing in text, values appearing in a database, or a schema element in one of the sources. In addition, the index needs to consider multiple ways of referring to the same real-world object. (Note that so far, research on reference reconciliation has focused on detecting when multiple references are about the same object).

Keeping the index up to date will be tricky, especially for participants that do not have mechanisms to notify it of updates. In addition, deciding which portions to cache in the local store and which indexes to build raises several interesting challenges in automated tuning.

4.5 Correctness guarantees

A core benefit of using a DSSP to access disparate data sources is the ability to do so with some confidence in the quality of the answers provided to queries and the effects and permanence of updates. Given the wide variance in administrative proximity and semantic integration (see Section 1.1) of the data sources in a dataspace, traditional DBMS guarantees for query answers and transactional updates will often be simply unobtainable. The research question then, is how to define realizable, practical, and meaningful levels of service guarantees that can be provided in a range of dataspaces. This challenge will require a rethinking of many fundamental data management principles, and the introduction of new abstractions. Tools to help designers and users understand the inherent tradeoffs in terms of quality, performance, and control will also be needed.

4.6 Theoretical foundations

There are several questions regarding the theoretical underpinnings of dataspaces. Clearly, there is need for a formal understanding of the different data models, relationships and answering queries in a dataspace. Digging deeper, in a traditional database theory, one of the main questions of interest is the expressive power of a query language. In the context of dataspaces, the analogous question would be the expressive power of a query language over a set of participants with certain properties on the relationships that are specified amongst them, i.e., what queries are expressible over a dataspace? Similarly, how can we detect semantically equivalent but syntactically different ways of answering queries?

5. Perspectives

To round out our discussion, we briefly discuss several important perspectives on dataspaces.

5.1 Relationship to Other Fields

Designing DSSPs builds on the traditional strengths our field and will involve significant extensions of data management techniques, but it will be crucial to leverage techniques from several other fields. We mention a few here. Recent developments in the field of knowledge representation (and the Semantic Web) offer two main benefits as we try to make sense of heterogeneous collections of data in a dataspace: simple but useful formalisms for representing ontologies, and the concept of URI (uniform resource identifiers) as a mechanism for referring to global constants on which there exists some agreement among multiple data providers. Similarly, as discussed earlier, several operations on dataspaces inherently involve some degree of uncertainty about the data, its lineage, correctness and completeness. The Uncertainty in AI Community had developed several formalisms for modeling uncertainty, but these tend to be very expressive. The challenge is to find models that are useful yet simple, understandable, and scalable.

Naturally, much of the data in a dataspace will be unstructured text. Hence, incorporating techniques from Information Retrieval will play a crucial role in building DSSP. Importantly, in a complex dataspace, users do not know exactly what they are looking for or how to interpret the results. Hence, it is important that they be able to effectively visualize results of searches and queries to better guide their exploration. Recent techniques from Information Visualization will be valuable here.

5.2 Teaching Dataspaces

An interesting litmus test for the concept of dataspaces is whether a course can be designed around it. Naturally, the foundations of dataspaces will evolve significantly as the research progresses, but we believe there is already sufficient material for a course. In addition to a review of basic data models and query languages, some of the topics that would be covered are: the challenges of heterogeneity and its different sources, architectures for data integration and data exchange, queries as a mechanism for data translation, algorithms for semi-automatic schema matching, different notions of QoS, supporting best-effort querying, and integrating structured and unstructured data querying. An important component of such a course would be to use and analyze successful examples of dataspaces (e.g., the Sloan Digital Sky Survey).

5.3 The Industrial Perspective

The concept of dataspaces is inspired in large part by challenges faced by industry today. In fact, there are many examples where industry is already making steps in this direction, but these steps are isolated from each other and there is clear need for a broader view that will yield a cleaner system abstraction and set of techniques.

For example, Enterprise Information Integration is starting to gain traction. The companies in this space are building systems to query multiple data sources within an organization. There are several examples of products that create indexes across multiple data sources for the purposes we mentioned above (e.g., Master Data Management, a component of NetWeaver of SAP). There are projects attempting to discover data sources within an enterprise, and there are quite a few companies looking at various aspects of enterprise meta-data management. Interestingly, the desktop search tools are also extending into the enterprise, coming from a completely different industry sector.

6. Conclusion

The most acute information management challenges within organizations today stem from the organizations' many diverse but often interrelated data sources. In this paper we have proposed the idea of dataspaces and the development of DataSpace Support Platforms (DSSP), as a means of addressing these challenges. DSSPs are intended to free application developers from having to continually re-implement basic data management functionality when dealing with complex, diverse, interrelated data sources, much in the same way that traditional DBMSs provide such leverage over structured relational databases. Unlike a DBMS, however, a DSSP does not assume complete control over the data in the dataspace. Instead, a DSSP allows the data to be managed by the participant systems, but provides a new set of services over the aggregate of the systems, while remaining sensitive to their requirements for autonomy.

Dataspaces can be seen as an umbrella for much of the research that is already being actively pursued in the database community; in fact this was one of our original goals. We have also, however, tried to outline several new research opportunities that arise from taking a more holistic view of emerging "data everywhere" challenges. These are challenges that the database research community is uniquely qualified to address, and we look forward to continued progress in extending the applicability of data management technology.

Acknowledgments

The authors were inspired (in different ways) by events at the CIDR 2005 Conference and the SIGMOD 2005 Post-PC Symposium to have the discussions that ultimately led to this document. Jennifer Widom played a key role in the original development of the dataspace concept. Serge Abiteboul, Phil Bernstein, Mike Carey, David DeWitt, Laura Haas, Zack Ives, Donald Kossmann, Mike Stonebraker, Dan Suciu, Jeff Ullman, Gerhard Weikum, and Stan Zdonik gave us useful advice (some of which we followed) on earlier instantiations of these ideas. We'd also like to thank the many colleagues who've read earlier (much longer) versions of this document.

References

[AAB+05] Serge Abiteboul, Rakesh Agrawal, Phil Bernstein, Mike Carey, Stefano Ceri, Bruce Croft, David DeWitt, Mike Franklin, Hector Garcia Molina, Dieter Gawlick, Jim Gray, Laura Haas, Alon Halevy, Joe Hellerstein, Yannis Ioannidis, Martin Kersten, Michael Pazzani, Mike Lesk, David Maier, Jeff Naughton, Hans Schek, Timos Sellis, Avi Silberschatz, Mike Stonebraker, Rick Snodgrass, Jeff Ullman, Gerhard Weikum, Jennifer Widom, and Stan Zdonik. The lowell database research self-assessment. *Commun. ACM*, 48(5):111–118, 2005.

[BBC+98] Phil Bernstein, Michael Brodie, Stefano Ceri, David DeWitt, Mike Franklin, Hector Garcia-Molina, Jim Gray, Jerry Held, Joe Hellerstein, H V Jagadish, Michael Lesk, Dave Maier, Jeff Naughton, Hamid Pirahesh, Mike Stonebraker, and Jeff Ullman. The asilomar report on database research. *ACM SIGMOD Record*, 27(4):74–80, 1998.

Scientific Data Management in the Coming Decade

Jim Gray, Microsoft
David T. Liu, Berkeley
Maria Nieto-Santisteban & Alex Szalay, Johns Hopkins University
David J. DeWitt, Wisconsin
Gerd Heber, Cornell
January 2005

An earlier version of this paper appeared Cyber Technology Watch, February 2005, http://www.ctwatch.org/quarterly/

Data-intensive science – a new paradigm

Scientific instruments and computer simulations are creating vast data stores that require new scientific methods to analyze and organize the data. Data volumes are approximately doubling each year. Since these new instruments have extraordinary precision, the data quality is also rapidly improving. Analyzing this data to find the subtle effects missed by previous studies requires algorithms that can simultaneously deal with huge datasets and that can find very subtle effects – finding both needles in the haystack and finding very small haystacks that were undetected in previous measurements.

The raw instrument and simulation data is processed by pipelines that produce standard data products. In the NASA terminology[1], the raw *Level 0* data is calibrated and rectified to *Level 1* datasets that are combined with other data to make derived *Level 2* datasets. Most analysis happens on these *Level 2* datasets with drill down to *Level 1* data when anomalies are investigated.

We believe that most new science happens when the data is examined in new ways. So our focus here is on data exploration, interactive data analysis, and integration of *Level 2* datasets.

Data analysis tools have not kept pace with our ability to capture and store data. Many scientists envy the pen-and-paper days when all their data used to fit in a notebook and analysis was done with a slide-rule. Things were simpler then; one could focus on the science rather than needing to be an information-technology-professional with expertise in arcane computer data analysis tools.

The largest data analysis gap is in this man-machine interface. How can we put the scientist back in control of his data? How can we build analysis tools that are intuitive and that augment the scientist's intellect rather than adding to the intellectual burden with a forest of arcane user tools? The real challenge is building this *smart notebook* that unlocks the data and makes it easy to capture, organize, analyze, visualize, and publish.

This article is about the data and data analysis layer within such a smart notebook. We argue that *the smart notebook* will access data presented by science centers that will provide the community with analysis tools and computational resources to explore huge data archives.

New data-analysis methods

The demand for tools and computational resources to perform scientific data-analysis is rising even faster than data volumes. This is a consequence of three phenomena: (1) More sophisticated algorithms consume more instructions to analyze each byte. (2) Many analysis algorithms are super-linear, often needing N^2 or N^3 time to process N data points. And (3) IO bandwidth has not kept pace with storage capacity. In the last decade, while capacity has grown more than 100-fold, storage bandwidth has improved only about 10-fold.

These three trends: algorithmic intensity, nonlinearity, and bandwidth-limits mean that the analysis is taking longer and longer. To ameliorate these problems, scientists will need better analysis algorithms that can handle extremely large datasets with approximate algorithms (ones with near-linear execution time) and they will need parallel algorithms that can apply many processors and many disks to the problem to meet cpu-density and bandwidth-density demands.

Science centers

These peta-scale datasets required a new work style. Today the typical scientist copies files to a local server and operates on the datasets using his own resources. Increasingly, the datasets are so large, and the application programs are so complex, that it is much more economical to move the end-user's programs to the data and only communicate questions and answers rather than moving the source data and its applications to the user's local system.

Science data centers that provide access to both the data and the applications that analyze the data are emerging as

[1] Committee on Data Management, Archiving, and Computing (CODMAC) Data Level Definitions http://science.hq.nasa.gov/research/earth_science_formats.html

service stations for one or another scientific domain. Each of these science centers curates one or more massive datasets, curates the applications that provide access to that dataset, and supports a staff that understands the data and indeed is constantly adding to and improving the dataset. One can see this with the SDSS at Fermilab, BaBar at SLAC, BIRN at SDSC, with Entrez-PubMed-GenBank at NCBI, and with many other datasets across other disciplines. These centers federate with others. For example BaBar has about 25 peer sites and CERN LHC expects to have many Tier1 peer sites. NCBI has several peers, and SDSS is part of the International Virtual Observatory.

The new work style in these scientific domains is to send questions to applications running at a data center and get back answers, rather than to bulk-copy raw data from the archive to your local server for further analysis. Indeed, there is an emerging trend to store a *personal workspace* (a *MyDB*) at the data center and deposit answers there. This minimizes data movement and allows collaboration among a group of scientists doing joint analysis. These personal workspaces are also a vehicle for data analysis groups to collaborate. Longer term, personal workspaces at the data center could become a vehicle for data publication – posting both the scientific results of an experiment or investigation along with the programs used to generate them in public read-only databases.

Many scientists will prefer doing much of their analysis at data centers because it will save them having to manage local data and computer farms. Some scientists may bring the small data extracts "home" for local processing, analysis and visualization – but it will be possible to do all the analysis at the data center using the personal workspace.

When a scientist wants to correlate data from two different data centers, then there is no option but to move part of the data from one place to another. If this is common, the two data centers will likely federate with one another to provide mutual data backup since the data traffic will justify making the copy.

Peta-scale data sets will require 1000-10,000 disks and thousands of compute nodes. At any one time some of the disks and some of the nodes will be broken. Such systems have to have a mechanism in place to protect against data loss, and provide availability even with a less than full configuration — a self-healing system is required. Replicating the data in science centers at different geographic locations is implied in the discussion above. Geographic replication provides both data availability and protects against data loss. Within a data center one can combine redundancy with a clever partitioning strategy to protect against failure at the disk controller or server level. While storing the data twice for redundancy, one can use different organizations (e.g. partition by space in one, and by time in the other) to optimize system performance. Failed can should be automatically recovered from the redundant copies with no interruption to database access, much as RAID5 disk arrays do today.

All these scenarios postulate easy data access, interchange and integration. Data must be self-describing in order to allow this. This self-description, or metadata, is central to all these scenarios; it enables generic tools to understand the data, and it enables people to understand the data.

Metadata enables data access

Metadata is the descriptive information about data that explains the measured attributes, their names, units, precision, accuracy, data layout and ideally a great deal more. Most importantly, metadata includes the data linage that describes how the data was measured, acquired or computed.

If the data is to be analyzed by generic tools, the tools need to "understand" the data. You cannot just present a bundle-of-bytes to a tool and expect the tool to intuit where the data values are and what they mean. The tool will want to know the metadata.

To take a simple example, given a file, you cannot say much about it – it could be anything. If I tell you it is a JPEG, you know it is a bitmap in http://www.jpeg.org/ format. JPEG files start with a header that describes the file layout, and often tells the camera, timestamp, and program that generated the picture. Many programs know how to read JPEG files and also produce new JPEG files that include metadata describing how the new image was produced. MP3 music files and PDF document files have similar roles – each is in a standard format, each carries some metadata, and each has an application suite to process and generate that file class.

If scientists are to read data collected by others, then the data must be carefully documented and must be published in forms that allow easy access and automated manipulation. In an ideal world there would be powerful tools that make it easy to capture, organize, analyze, visualize, and publish data. The tools would do data mining and machine learning on the data, and would make it easy to script workflows that analyze the data. Good metadata for the inputs is essential to make these tools automatic. Preserving and augmenting this metadata as part of the processing (data lineage) will be a key benefit of the next-generation tools.

All the derived data that the scientist produces must also be carefully documented and published in forms that

allow easy access. Ideally much of this metadata would be automatically generated and managed as part of the workflow, reducing the scientist's intellectual burden.

Semantic convergence: numbers to objects

Much science data is in the form of numeric arrays generated by instruments and simulations. Simple and convenient data models have evolved to represent arrays and relationships among them. These data models can also represent data lineage and other metadata by including narrative text, data definitions, and data tables within the file. HDF[2], NetCDF[3] and FITS[4] are good examples of such standards. They each include a library that encapsulates the files and provides a platform-independent way to read sub-arrays and to create or update files. Each standard allows easy data interchange among scientists. Generic tools that analyze and visualize these higher-level file formats are built atop each of these standards.

While the commercial world has standardized on the relational data model and SQL, no single standard or tool has critical mass in the scientific community. There are many parallel and competing efforts to build these tool suites – at least one per discipline. Data interchange outside each group is problematic. In the next decade, as data interchange among scientific disciplines becomes increasingly important, a common HDF-like format and package for all the sciences will likely emerge.

Definitions of common terminology (units and measurements) are emerging within each discipline. We are most familiar with the Universal Content Descriptors (UCD[5]) of the Astronomy community that define about a thousand core astrophysics units, measurements, and concepts. Almost every discipline has an analogous ontology (a.k.a., *controlled vocabulary*) effort. These efforts will likely start to converge over the next decade – probably as part of the converged format standard. This will greatly facilitate tool-building and tools since an agreement on these concepts can help guide analysis tool designs.

In addition to standardization, computer-usable ontologies will help build the Semantic Web: applications will be semantically compatible beyond the mere syntactic compatibility that current-generation of Web services offer with type matching interfaces. However, it will take some time before high-performance general-purpose *ontology engines* will be available and integrated with data analysis tools.

[2] http://hdf.ncsa.uiuc.edu/HDF5/
[3] http://my.unidata.ucar.edu/content/software/netcdf/
[4] http://fits.gsfc.nasa.gov/
[5] http://vizier.u-strasbg.fr/doc/UCD.htx

Database users on the other hand are well positioned to prototype such applications: a database schema, though not a complete ontology in itself, can be a rich ontology extract. SQL can be used to implement a rudimentary *semantic algebra*. The XML integration in modern Database Management Systems (DBMS) opens the door for existing standards like RDF and OWL.

Visualization or better *visual exploration* is a prime example of an application where success is determined by the ability to map a question formulated in the conceptual framework of the domain ontology onto the querying capabilities of a (meta-) data analysis backend. For the time being, a hybrid of SQL and XQuery is the only language suitable to serve as the target assembly language in this translation process.

Metadata enables data independence

The separation of data and programs is artificial – one cannot see the data without using a program and most programs are data driven. So, it is paradoxical that the data management community has worked for 40 years to achieve something called *data independence* – a clear separation of programs from data. Database systems provide two forms of data independence termed *physical data independence* and *logical data independence*.

Physical data independence comes in many different forms. However, in all cases the goal is to be able to change the underlying physical data organization without breaking any application programs that depend on the old data format. One example of physical data independence is the ability of a database system to partition the rows of a table across multiple disks and/or multiple nodes of a cluster without requiring that any application programs be modified. The mapping of the fields of each row of a relational table to different disks is another important example of physical data independence. While a database system might choose to map each row to a contiguous storage container (e.g. a record) on a single disk page, it might also choose to store large, possibly infrequently referenced attributes of a table corresponding to large text objects, JPEG images, or multidimensional arrays in separate storage containers on different disk pages and/or different storage volumes in order to maximize the overall performance of the system. Again, such physical storage optimizations are implemented to be completely transparent to application programs except, perhaps, for a change in their performance. In the scientific domain the analogy would be that you could take a working application program that uses a C struct to describe its data records on disk and change the physical layout of the records without having to rewrite or even recompile the application program (or any of the other application

programs that access the same data). By allowing such techniques, physical data independence allows performance improvements by reorganizing data for parallelism–at little or no extra effort on the part of scientists.

Modern database systems also provide *logical data independence* that insulates programs from changes to the logical database design – allowing designers to add or delete relationships and to add information to the database. While physical data independence is used to hide changes in the physical data organizations, logical data independence hides changes in the logical organization of the data. Logical data independence is typically supported using *views*. A view defines a virtual table that is specified using a SQL query over one or more base tables and/or other views. Views serve many purposes including increased security (by hiding attributes from applications and/or users without a legitimate need for access) and enhanced performance (by materializing views defined by complex SQL queries over very large input tables). But views are primarily used to allow old programs to operate correctly even as the underlying database is reorganized and redesigned. For example, consider a program whose correct operation depends on some table T that a database administrator wants to reorganize by dividing vertically into two pieces stored in tables T' and T". To preserve applications that depend on T, the database administrator can then define a view over T' and T" corresponding to the original definition of table T, allowing old programs to continue to operate correctly.

In addition, data evolves. Systems evolve from EBCDIC to ASCII to Unicode, from proprietary-float to IEEE-float, from marks to euros, and from 8-character ASCII names to 1,000 character Unicode names. It is important to be able to make these changes without breaking the millions of lines of existing programs that want to see the data in the old way. Views are used to solve these problems by dynamically translating data to the appropriate formats (converting among character and number representations, converting among 6-digit and 9-digit postal codes, converting between long-and-short names, and hiding new information from old programs.) The pain of the Y2K (converting from 2-character to 4-character years) taught most organizations the importance of data independence.

Database systems use a *schema* to implement both logical and physical data independence. The schema for a database holds all metadata including table and view definitions as well as information on what indices exist and how tables are mapped to storage volumes (and nodes in a parallel database environment). Separating the data and the metadata from the programs that manipulate the data is crucial to data independence. Otherwise, it is essentially impossible for other programs to find the metadata which, in turn, makes it essentially impossible for multiple programs to share a common database. Object-oriented programming concepts have refined the separation of programs and data. Data classes encapsulated with methods provide data independence and make it much easier to evolve the data without perturbing programs. So, these ideas are still evolving.

But the key point of this section is that an explicit and standard data access layer with precise metadata and explicit data access is essential for data independence.

Set-oriented data access gives parallelism

As mentioned earlier, scientists often start with numeric data arrays from their instruments or simulations. Often, these arrays are accompanied by tabular data describing the experimental setup, simulation parameters, or environmental conditions. The data are also accompanied by documents that explain the data.

Many operations take these arrays and produce new arrays, but eventually, the arrays undergo *feature extraction* to produce *objects* that are the basis for further analysis. For example, raw astronomy data is converted to object catalogs of stars and galaxies. Stream-gauge measurements are converted to stream-flow and water-quality time-series data, serum-mass-spectrograms are converted to records describing peptide and protein concentrations, and raw high-energy physics data are converted to events.

Most scientific studies involve exploring and data mining these object-oriented tabular datasets. The scientific file-formats of HDF, NetCDF, and FITS can represent tabular data but they provide minimal tools for searching and analyzing tabular data. Their main focus is getting the tables and sub-arrays into your Fortran/C/Java/Python address space where you can manipulate the data using the programming language.

This Fortran/C/Java/Python file-at-a-time procedural data analysis is nearing the breaking point. The data avalanche is creating billions of files and trillions of events. The file-oriented approach postulates that files are organized into directories. The directories relate all data from some instrument or some month or some region or some laboratory. As things evolve, the directories become hierarchical. In this model, data analysis proceeds by searching all the relevant files – opening each file, extracting the relevant data and then moving onto the next file. When all the relevant data has been gathered in memory (or in intermediate files) the program can begin its analysis. Performing this *filter-then-analyze*, data

analysis on large datasets with conventional procedural tools runs slower and slower as data volumes increase. Usually, they use only one-cpu-at-a-time; one-disk-at-a-time and they do a brute-force search of the data. Scientists need a way (1) to use intelligent indices and data organizations to subset the search, (2) to use parallel processing and data access to search huge datasets within seconds, and (3) to have powerful analysis tools that they can apply to the subset of data being analyzed.

One approach to this is to use the MPI (Message Passing Interface) parallel programming environment to write procedural programs that stream files across a processor array – each node of the array exploring one part of the hierarchy. This is adequate for highly-regular array processing tasks, but it seems too daunting for ad-hoc analysis of tabular data. MPI and the various array file formats lack indexing methods other than partitioned sequential scan. MPI itself lacks any notion of metadata beyond file names.

As file systems grow to petabyte-scale archives with billions of files, the science community must create a synthesis of database systems and file systems. At a minimum, the file hierarchy will be replaced with a database that catalogs the attributes and lineage of each file. Set-oriented file processing will make file names increasingly irrelevant – analysis will be applied to "all data with these attributes" rather than working on a list of file/directory names or name patterns. Indeed, the files themselves may become irrelevant (they are just containers for data.) One can see a harbinger of this idea in the Map-Reduce approach pioneered by Google[6]. From our perspective, the key aspect of Google Map-Reduce is that it applies thousands of processors and disks to explore large datasets in parallel. That system has a very simple data model appropriate for the Google processing, but we imagine it could evolve over the next decade to be quite general.

The database community has provided automatic query processing along with CPU and IO parallelism for over two decades. Indeed, this automatic parallelism allows large corporations to mine 100-Terabyte datasets today using 1000 processor clusters. We believe that many of those techniques apply to scientific datasets[7].

Other useful database features

Database systems are also approaching the peta-scale data management problem driven largely by the need to manage huge information stores for the commercial and governmental sectors. They hide the file concept and deal with data collections. They can federate many different sources letting the program view them all as a single data collection. They also let the program pivot on any data attributes.

Database systems provide very powerful data definition tools to specify the abstract data formats and also specify how the data is organized. They routinely allow the data to be replicated so that it can be organized in several ways (by time, by space, by other attributes). These techniques have evolved from mere indices to materialized views that can combine data from many sources.

Database systems provide powerful associative search (search by value rather than by location) and provide automatic parallel access and execution essential to peta-scale data analysis. They provide non-procedural and parallel data search to quickly find data subsets, and a many tools to automate data design and management.

In addition, data analysis using data cubes has made huge advances, and now efforts are focused on integrating machine learning algorithms that infer trends, do data clustering, and detect anomalies. All these tools are aimed at making it easy to analyze commercial data, but they are equally applicable to scientific data analysis.

Ending the impedance mismatch

Conventional tabular database systems are adequate for analyzing objects (galaxies, spectra, proteins, events, etc.). But even there, the support for time-sequence, spatial, text and other data types is often awkward. Database systems have not traditionally supported science's core data type: the N-dimensional array. Arrays have had to masquerade as blobs (binary large objects) in most systems. This collection of problems is generally called the *impedance mismatch* – meaning the mismatch between the programming model and the database capabilities. The impedance mismatch has made it difficult to map many science applications into conventional tabular database systems.

But, database systems are changing. They are being integrated with programming languages so that they can support object-oriented databases. This new generation of object relational database systems treats any data type (be it a native float, an array, a string, or a compound object like an XML or HTML document) as an encapsulated type that can be stored as a value in a field of a record. Actually, these systems allow the values to be either stored directly in the record (embedded) or to be pointed to by the record (linked). This linking-embedding object model nicely accommodates the integration of database

[6] "MapReduce: Simplified Data Processing on Large Clusters," J. Dean, S. Ghemawat, ACM OSDI, Dec. 2004.

[7] "Parallel Database Systems: the Future of High Performance Database Systems", D. DeWitt, J. Gray, CACM, Vol. 35, No. 6, June 1992.

systems and file systems – files are treated as linked-objects. Queries can read and write these extended types using the same techniques they use on native types. Indeed we expect HDF and other file formats to be added as types to most database systems.

Once you can put your types and your programs inside the database you get the parallelism, non-procedural query, and data independence advantages of traditional database systems. We believe this database, file system, and programming language integration will be the key to managing and accessing peta-scale data management systems in the future.

What's wrong with files?

Everything builds from files as a base. HDF uses files. Database systems use files. But, file systems have no metadata beyond a hierarchical directory structure and file names. They encourage a do-it-yourself- data-model that will not benefit from the growing suite of data analysis tools. They encourage do-it-yourself-access-methods that will not do parallel, associative, temporal, or spatial search. They also lack a high-level query language. Lastly, most file systems can manage millions of files, but by the time a file system can deal with billions of files, it has become a database system.

As you can see, we take an ecumenical view of what a database is. We see NetCDF, HDF, FITS, and Google Map-Reduce as nascent database systems (others might think of them as file systems). They have a schema language (metadata) to define the metadata. They have a few indexing strategies, and a simple data manipulation language. They have the start of non-procedural and parallel programming. And, they have a collection of tools to create, access, search, and visualize the data. So, in our view they are simple database systems.

Why scientists don't use databases today

Traditional database systems have lagged in supporting core scientific data types but they have a few things scientists desperately need for their data analysis: non-procedural query analysis, automatic parallelism, and sophisticated tools for associative, temporal, and spatial search.

If one takes the controversial view that HDF, NetCDF, FITS, and Root are nascent database systems that provide metadata and portability but lack non-procedural query analysis, automatic parallelism, and sophisticated indexing, then one can see a fairly clear path that integrates these communities.

Some scientists use databases for some of their work, but as a general rule, most scientists do not. Why? Why are tabular databases so successful in commercial applications and such a flop in most scientific applications? Scientific colleagues give one or more of the following answers when asked why they do not use databases to manage their data:

- We don't see any benefit in them. The cost of learning the tools (data definition and data loading, and query) doesn't seem worth it.
- They do not offer good visualization/plotting tools.
- I can handle my data volumes with my programming language.
- They do not support our data types (arrays, spatial, text, etc.).
- They do not support our access patterns (spatial, temporal, etc.).
- We tried them but they were too slow.
- We tried them but once we loaded our data we could no longer manipulate the data using our standard application programs.
- They require an expensive guru (database administrator) to use.

All these answers are based on experience and considerable investment. Often the experience was with older systems (a 1990 vintage database system) or with a young system (an early object-oriented database or an early version of Postgres or MySQL.) Nonetheless, there is considerable evidence that databases have to improve a lot before they are worth a second look.

Why things are different now

The thing that forces a second look now is that the file-ftp *modus operandi* just will not work for peta-scale datasets. Some new way of managing and accessing information is needed. We argued that metadata is the key to this and that a non-procedural data manipulation language combined with data indexing is essential to being able to search and analyze the data.

There is a convergence of file systems, database systems, and programming languages. Extensible database systems use object-oriented techniques from programming languages to allow you to define complex objects as native database types. Files (or extended files like HDF) then become part of the database and benefit from the parallel search and metadata management. It seems very likely that these nascent database systems will be integrated with the main-line database systems in the next decade or that some new species of metadata driven analysis and workflow system will supplant both traditional databases and the science-specific file formats and their tool suites.

Some hints of success

There are early signs that this is a good approach. One of us has shown that the doing analysis atop a database system is vastly simpler and runs much faster than the corresponding file-oriented approach[8]. The speedup is due to better indexing and parallelism.

We have also had considerable success in adding user defined functions and stored procedures to astronomy databases. The MyDB and CasJobs work for the Sloan Digital Sky Survey give a good example of moving-programs-to-the-database[9].

The BaBar experiments at SLAC manage a petabyte store of event data. The system uses a combination of Oracle to manage some of the file archive and also a physics-specific data analysis system called Root for data analysis[10].

The GridDB[11] workflow system at UC Berkeley expands the role of database systems into pipeline processing, a domain traditionally serviced by "process-centric" middlewares[12,13]. Process-centric middlewares automatically parallelize workflows of imperative, file-based programs (e.g. those written in Fortran/C/Python/Java) by making use of a "workflow schema", which describes programs and their dependencies. GridDB uses database techniques to improve pipeline processing; specifically, it uses schemas that not only contain workflow information, but also incorporate data information (i.e. a database schema). The combination of both workflow and data schemas enable declarative interfaces, and improves the interactivity and performance of pipeline processing.

Adaptive Finite Element simulations spend considerable time and programming effort on input, output, and checkpointing. We (Heber) use a database to represent large Finite Element models. The initial model is represented in the database and each checkpoint and analysis step is written to the database. Using a database allows queries to define more sophisticated mesh partitions and allows concurrent indexed access to the simulation data for visualization and computational steering. Commercial Finite Element packages each use a proprietary form of a "database". They are, however, limited in scope, functionality, and scalability, and are typically buried inside the particular application stack. Each worker in the MPI job gets its partition from the database (as a query) and dumps its progress to the database. These dumps are two to four orders of magnitude larger than the input mesh and represent a performance challenge in both traditional and database environments. The database approach has the added benefit that visualization tools can watch and steer the computation by reading and writing the database. Finally, while we have focused on the ability of databases to simplify and speedup the production of raw simulation data, we cannot understate its core competency: providing declarative data analysis interfaces. It is with these tools that scientists spend most of their time. We hope to apply similar concepts to some turbulence studies being done at Johns Hopkins.

Summary

Science centers that curate and serve science data are emerging around next-generation science instruments. The world-wide telescope, GenBank, and the BaBar collaborations are prototypes of this trend. One group of scientists is collecting the data and managing these archives. A larger group of scientists are exploring these archives the way previous generations explored their private data. Often the results of the analysis are fed back to the archive to add to the corpus.

Because data collection is now separated from data analysis, extensive metadata describing the data in standard terms is needed so people and programs can understand the data. Good metadata becomes central for data sharing among different disciplines and for data analysis and visualization tools.

There is a convergence of the nascent-databases (HDF, NetCDF, FITS,..) which focus primarily on the metadata issues and data interchange, and the traditional data management systems (SQL and others) that have focused on managing and analyzing very large datasets. The traditional systems have the virtues of automatic parallelism, indexing, and non-procedural access, but they need to embrace the data types of the science community and need to co-exist with data in file systems. We believe the emphasis on extending database systems by unifying databases with programming languages so that

[8] "When Database Systems Meet the Grid," M. Nieto Santisteban et. al., CIDR, 2005, http://www-db.cs.wisc.edu/cidr/papers/P13.pdf

[9] "Batch is back: CasJobs serving multi-TB data on the Web," W. O'Mullane, et. al, in preparation.

[10] "Lessons Learned from Managing a Petabyte," J. Becla and D. L. Wang, CIDR, 2005, http://www-db.cs.wisc.edu/cidr/papers/P06.pdf

[11] D. T. Liu and M. J. Franklin, VLDB, 2004, www.cs.berkeley.edu/~dtliu/pubs/griddb_vldb04.pdf

[12] M. Litzkow, M. Livny and M. Mutka, Condor - A Hunter of Idle Workstations, International Conference of Distributed Computing Systems, 1988

[13] I. Foster and C. Kesselman, Globus: A Metacomputing Infrastructure Toolkit, Journal of Supercomputer Applications and High Performance Computing, 1997

one can either embed or link new object types into the data management system will enable this synthesis.

Three technical advances will be crucial to scientific analysis: (1) extensive metadata and metadata standards that will make it easy to discover what data exits, make it easy for people and programs to understand the data, and make it easy to track data lineage; (2) great analysis tools that allow scientists to easily ask questions, and to easily understand and visualize the answers; and (3) set-oriented data parallelism access supported by new indexing schemes and new algorithms that allow us to interactively explore peta-scale datasets.

The goal is a *smart notebook* that empowers scientists to explore the world's data. Science data centers with computational resources to explore huge data archives will be central to enabling such notebooks. Because data is so large, and IO bandwidth is not keeping pace, moving code to data will be essential to performance. Consequently, science centers will remain the core vehicle and federations will likely be secondary. Science centers will provide both the archives and the institutional infrastructure to develop these peta-scale archives and the algorithms and tools to analyze them.

The 8 Requirements of Real-Time Stream Processing

Michael Stonebraker
Computer Science and Artificial Intelligence Laboratory, M.I.T., and StreamBase Systems, Inc.
stonebraker@csail.mit.edu

Uğur Çetintemel
Department of Computer Science, Brown University, and StreamBase Systems, Inc.
ugur@cs.brown.edu

Stan Zdonik
Department of Computer Science, Brown University, and StreamBase Systems, Inc.
sbz@cs.brown.edu

ABSTRACT

Applications that require real-time processing of high-volume data steams are pushing the limits of traditional data processing infrastructures. These stream-based applications include market feed processing and electronic trading on Wall Street, network and infrastructure monitoring, fraud detection, and command and control in military environments. Furthermore, as the "sea change" caused by cheap micro-sensor technology takes hold, we expect to see everything of material significance on the planet get "sensor-tagged" and report its state or location in real time. This sensorization of the real world will lead to a "green field" of novel monitoring and control applications with high-volume and low-latency processing requirements.

Recently, several technologies have emerged—including off-the-shelf stream processing engines—specifically to address the challenges of processing high-volume, real-time data without requiring the use of custom code. At the same time, some existing software technologies, such as main memory DBMSs and rule engines, are also being "repurposed" by marketing departments to address these applications.

In this paper, we outline eight requirements that a system software should meet to excel at a variety of real-time stream processing applications. Our goal is to provide high-level guidance to information technologists so that they will know what to look for when evaluation alternative stream processing solutions. As such, this paper serves a purpose comparable to the requirements papers in relational DBMSs and on-line analytical processing. We also briefly review alternative system software technologies in the context of our requirements.

The paper attempts to be vendor neutral, so no specific commercial products are mentioned.

1. INTRODUCTION

On Wall Street and other global exchanges, electronic trading volumes are growing exponentially. Market data feeds can generate tens of thousands of messages per second. The Options Price Reporting Authority (OPRA), which aggregates all the quotes and trades from the options exchanges, estimates peak rates of 122,000 messages per second in 2005, with rates doubling every year [13]. This dramatic escalation in feed volumes is stressing or breaking traditional feed processing systems. Furthermore, in electronic trading, a latency of even one second is unacceptable, and the trading operation whose engine produces the most current results will maximize arbitrage profits. This fact is causing financial services companies to require very high-volume processing of feed data with very low latency.

Similar requirements are present in monitoring computer networks for denial of service and other kinds of security attacks. Real-time fraud detection in diverse areas from financial services networks to cell phone networks exhibits similar characteristics. In time, process control and automation of industrial facilities, ranging from oil refineries to corn flakes factories, will also move to such "firehose" data volumes and sub-second latency requirements.

There is a "sea change" arising from the advances in micro-sensor technologies. Although RFID has gotten the most press recently, there are a variety of other technologies with various price points, capabilities, and footprints (e.g., mote [1] and Lojack [2]). Over time, this sea change will cause everything of material significance to be sensor-tagged to report its location and/or state in real time.

Military has been an early driver and adopter of wireless sensor network technologies. For example, the US Army has been investigating putting vital-signs monitors on all soldiers. In addition, there is already a GPS system in many military vehicles, but it is not connected yet into a closed-loop system. Using this technology, the army would like to monitor the position of all vehicles and determine, in real time, if they are off course.

Other sensor-based monitoring applications will come over time in non-military domains. Tagging will be applied to customers at amusement parks for ride management and prevention of lost children. More sophisticated "easy-pass" systems will allow congestion-based tolling of automobiles on freeways (which was the inspiration behind the Linear Road Benchmark [5]) as well as optimized routing of cars in a metropolitan area. The processing of "firehoses" of real-time data from existing and newly-emerging monitoring applications presents a major stream processing challenge and opportunity.

Traditionally, custom coding has been used to solve high-volume, low-latency streaming processing problems. Even though the "roll your own" approach is universally despised because of its inflexibility, high cost of development and maintenance, and slow response time to new feature requests, application developers had to resort to it as they have not had good luck with traditional off-the-shelf system software.

Recently, several traditional system software technologies, such as main memory DBMSs and rule engines, have been repurposed and remarketed to address this application space. In addition, Stream Processing Engines (e.g., Aurora [8], STREAM [4], TelegraphCQ [9]), a new class of system software, have emerged to specifically support high-volume, low-latency stream processing applications.

In this paper, we describe eight characteristics that a system software must exhibit to excel at a variety of real-time stream processing applications. Our goal is to provide information technologists high-level guidance so that they know what to look for when evaluating their options. Thus, this paper shares a similar goal with earlier papers that present requirements for relational DBSMs [10, 11] and on-line analytical processing [12].

We present these features as a collection of eight rules in the next section. We then review the alternative technologies and summarize how they measure up for real-time stream processing in Section 3. We conclude with final remarks in Section 4.

2. EIGHT RULES FOR STREAM PROCESSING

Rule 1: Keep the Data Moving

To achieve low latency, a system must be able to perform message processing without having a costly storage operation in the critical processing path. A storage operation adds a great deal of unnecessary latency to the process (e.g., committing a database record requires a disk write of a log record). For many stream processing applications, it is neither acceptable nor necessary to require such a time-intensive operation before message processing can occur. Instead, messages should be processed "in-stream" as they fly by. See Figure 1 for an architectural illustration of this *straight-through processing* paradigm.

An additional latency problem exists with systems that are *passive*, as such systems wait to be told what to do by an application before initiating processing. Passive systems require applications to continuously *poll* for conditions of interest. Unfortunately, polling results in additional overhead on the system as well as the application, and additional latency, because (on average) half the polling interval is added to the processing delay. Active systems avoid this overhead by incorporating built-in event/data-driven processing capabilities.

> *The first requirement for a real-time stream processing system is to process messages "in-stream", without any requirement to store them to perform any operation or sequence of operations. Ideally the system should also use an active (i.e., non-polling) processing model.*

Rule 2: Query using SQL on Streams (StreamSQL)

In streaming applications, some querying mechanism must be used to find output events of interest or compute real-time analytics. Historically, for streaming applications, general purpose languages such as C++ or Java have been used as the workhorse development and programming tools. Unfortunately, relying on low-level programming schemes results in long development cycles and high maintenance costs.

In contrast, it is very much desirable to process moving real-time data using a high-level language such as SQL. SQL has remained the most enduring standard database language over three decades. SQL's success at expressing complex data transformations derives from the fact that it is based on a set of very powerful data processing primitives that do filtering, merging, correlation, and aggregation. SQL is explicit about how these primitives interact

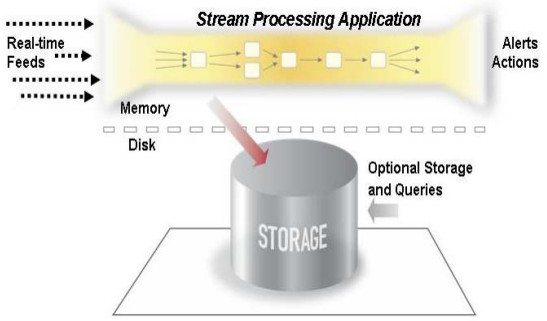


Figure 1: "Straight-through" processing of messages with optional storage.

so that its meaning can be easily understood independently from runtime conditions. Furthermore, SQL is a widely promulgated standard that is understood by hundreds of thousands of database programmers and is implemented by every serious DBMS in commercial use today, due to its combination of functionality, power, and relative ease-of-use. Given that millions of relational database servers running SQL are already installed and operating globally today, it makes good sense to leverage the familiar SQL querying model and operators, and simply extend them to perform processing on continuous data streams.

In order to address the unique requirements of stream processing, StreamSQL, a variant of the SQL language specifically designed to express processing on continuous streams of data, is needed. StreamSQL should extend the semantics of standard SQL (that assumes records in a finite stored dataset) by adding to it *rich* windowing constructs and stream-specific operators.

Although a traditional SQL system knows it is finished computing when it gets to the end of a table, because streaming data never ends, a stream processing engine must be instructed when to finish such an operation and output an answer. The *window* construct serves this purpose by defining the "scope" of a multi-message operator such as an aggregate or a join.

Windows should be definable over time (probably the most common usage case), number of messages, or breakpoints in other attributes in a message. Such windows should be able to *slide* a variable amount from the current window (e.g., a window could be five ticks wide and the next window could slide by one tick from the current one). As a result, depending on the choice of window size and slide parameters, windows can be made disjoint or overlapping. A sliding window example is shown in Figure 2.

Furthermore, new stream-oriented operators that are not present in the standard SQL are needed. An example is a "Merge" operator that multiplexes messages from multiple streams in a manner that

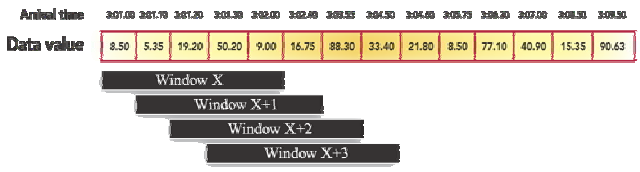


Figure 2: Windows define the scope of operations. The window has a size of 4 messages and slides by 1 each time the associated operator is executed. Consecutive windows overlap.

is sensitive to arrival times and ordering of data messages.
Finally, the operator set must be extensible, so that developers can easily achieve new processing functionality within the system (e.g., to implement a proprietary analysis algorithm on the streaming data).

The second requirement is to support a high-level "StreamSQL" language with built-in extensible stream-oriented primitives and operators.

Rule 3: Handle Stream Imperfections (Delayed, Missing and Out-of-Order Data)

In a conventional database, data is always present before it is queried against, but in a real-time system, since the data is never stored, the infrastructure must make provision for handling data that is late or delayed, missing, or out-of-sequence.

One requirement here is the ability to time out individual calculations or computations. For example, consider a simple real-time business analytic that computes the average price of the last tick for a collection of 25 securities. One need only wait for a tick from each security and then output the average price. However, suppose one of the 25 stocks is thinly traded, and no tick for that symbol will be received for the next 10 minutes. This is an example of a computation that must *block*, waiting for input to complete its calculation. Such input may or may not arrive in a timely fashion. In fact, if the SEC orders a stop to trading in one of the 25 securities, then the calculation will block indefinitely.

In a real-time system, it is *never* a good idea to allow a program to wait indefinitely. Hence, every calculation that can block must be allowed to *time out*, so that the application can continue with partial data. Any real-time processing system must have such time-outs for any potentially blocking operation.

Dealing with out-of-order data introduces similar challenges. Ordinarily, a time window (e.g., [9:00 – 9:01]) would be closed once a message with a timestamp greater than the window's closing time is received. However, such an action assumes that the data arrives in timestamp order, which may not be the case. To deal with out-of-order data, a mechanism must be provided to allow windows to stay open for an additional period of time. One solution specified in Aurora was the notion of *slack* [3].

The third requirement is to have built-in mechanisms to provide resiliency against stream "imperfections", including missing and out-of-order data, which are commonly present in real-world data streams.

Rule 4: Generate Predictable Outcomes

A stream processing system must process time-series messages in a predictable manner to ensure that the results of processing are deterministic and repeatable.

For example, consider two feeds, one containing TICKS data with fields:

TICKS (stock_symbol, volume, price, time),

and the other a SPLITS feed, which indicates when a stock splits, with the format:

SPLITS (symbol, time, split_factor).

A typical stream processing application would be to produce the real-time split-adjusted price for a collection of stocks. The price must be adjusted for the cumulative split_factor that has been seen. The correct answer to this computation can be produced when messages are processed by the system in ascending time order, regardless of when the messages arrive to the system. If a split message is processed out-of-order, then the split-adjusted price for the stock in question will be wrong for one or more ticks. Notice that it is insufficient to simply sort-order messages before they are input to the system—correctness can be guaranteed only if time-ordered, deterministic processing is maintained throughout the entire processing pipeline.

The ability to produce predictable results is also important from the perspective of fault tolerance and recovery, as replaying and reprocessing the same input stream should yield the same outcome regardless of the time of execution.

The fourth requirement is that a stream processing engine must guarantee predictable and repeatable outcomes.

Rule 5: Integrate Stored and Streaming Data

For many stream processing applications, comparing "present" with "past" is a common task. Thus, a stream processing system must also provide for careful management of stored state. For example, in on-line data mining applications (such as detecting credit card or other transactional fraud), identifying whether an activity is "unusual" requires, by definition, gathering the usual activity patterns over time, summarizing them as a "signature", and comparing them to the present activity in real time. To realize this task, both historical and live data need to be integrated within the same application for comparison.

A very popular extension of this requirement comes from firms with electronic trading applications, who want to write a trading algorithm and then test it on historical data to see how it would have performed and to test alternative scenarios. When the algorithm works well on historical data, the customer wants to switch it over to a live feed *seamlessly*; i.e., without modifying the application code. Seamless switching ensures that new errors are not introduced by changes to the program.

Another reason for seamless switching is the desire to compute some sort of business analytic starting from a past point in time (such as starting two hours ago), "catch up" to real time, and then seamlessly continue with the calculation on live data. This capability requires switching automatically from historical to live data, without the manual intervention of a human.

For low-latency streaming data applications, interfacing with a client-server database connection to efficiently store and access persistent state will add excessive latency and overhead to the application. Therefore, state must be stored in the same operating system address space as the application using an embedded database system. Therefore, the scope of a StreamSQL command should be either a live stream or a stored table in the embedded database system.

The fifth requirement is that a stream processing system should have the capability to efficiently store, access, and modify state information, and combine it with live streaming data. For seamless integration, the system should use a uniform language when dealing with either type of data.

Rule 6: Guarantee Data Safety and Availability

To preserve the integrity of mission-critical information and avoid disruptions in real-time processing, a stream processing system must use a high-availability (HA) solution.

High availability is a critical concern for most stream processing applications. For example, virtually all financial services firms expect their applications to stay up all the time, no matter what happens. If a failure occurs, the application needs to failover to

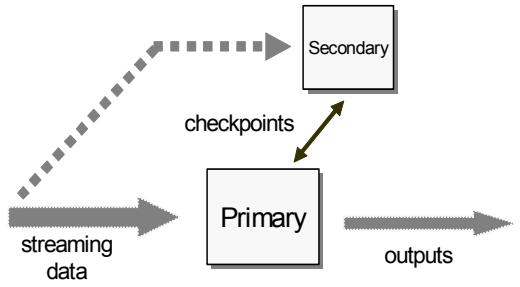


Figure 3: "Tandem-style" hot backup and failover can ensure high availability for real-time stream processing.

backup hardware and keep going. Restarting the operating system and recovering the application from a log incur too much overhead and is thus not acceptable for real-time processing. Hence, a "Tandem-style" hot backup and real-time failover scheme [6], whereby a secondary system frequently synchronizes its processing state with a primary and takes over when primary fails, is the best reasonable alternative for these types of applications. This HA model is shown in Figure 3.

The sixth requirement is to ensure that the applications are up and available, and the integrity of the data maintained at all times, despite failures.

Rule 7: Partition and Scale Applications Automatically

Distributed operation is becoming increasingly important given the favorable price-performance characteristics of low-cost commodity clusters. As such, it should be possible to split an application over multiple machines for scalability (as the volume of input streams or the complexity of processing increases), without the developer having to write low-level code.

Stream processing systems should also support multi-threaded operation to take advantage of modern multi-processor (or multi-core) computer architectures. Even on a single-processor machine, multi-threaded operation should be supported to avoid blocking for external events, thereby facilitating low latency.

Not only must scalability be provided easily over any number of machines, but the resulting application should automatically and transparently load-balance over the available machines, so that the application does not get bogged down by a single overloaded machine.

The seventh requirement is that a stream processing system must be able to distribute its processing across multiple processors and machines to achieve incremental scalability. Ideally, the distribution should be automatic and transparent.

Rule 8: Process and Respond Instantaneously

None of the preceding rules will make any difference alone unless an application can "keep up", i.e., process high-volumes of streaming data with very low latency. In numbers, this means capability to process tens to hundreds of thousands of messages per second with latency in the microsecond to millisecond range on top of COTS hardware.

To achieve such high performance, the system should have a highly-optimized execution path that minimizes the ratio of overhead to useful work. As exemplified by the previous rules, a critical issue here is to minimize the number of "boundary crossings" by integrating all critical functionality (e.g., processing and storage) into a single system process. However, this is not sufficient by itself; all system components need to be designed with high performance in mind.

To make sure that a system can meet this requirement, it is imperative that any user with a high-volume streaming application carefully test any product he might consider for throughput and latency on his target workload.

The eighth requirement is that a stream processing system must have a highly-optimized, minimal-overhead execution engine to deliver real-time response for high-volume applications.

3. SYSTEM SOFTWARE TECHNOLOGIES for STREAM PROCESSING

3.1 Basic Architectures

In addition to custom coding, there are at least three different software system technologies that can potentially be applied to solve high-volume low-latency streaming problems. These are DBMSs, rule engines, and stream processing engines, which we discuss below:

- **Database Management Systems (DBMSs)** are widely used due to their ability to reliably store large data sets and efficiently process human-initiated queries. Main-memory DBMSs can provide higher performance than traditional DBMSs by avoiding going to disk for most operations, given sufficient main memory.

  Figure 4(i) illustrates the basic DBMS architecture. Streaming data is entered into the DBMS directly or through a loading application. A collection of applications can then manipulate DBMS data. A client can use these pre-built applications, often with run-time arguments, and can also

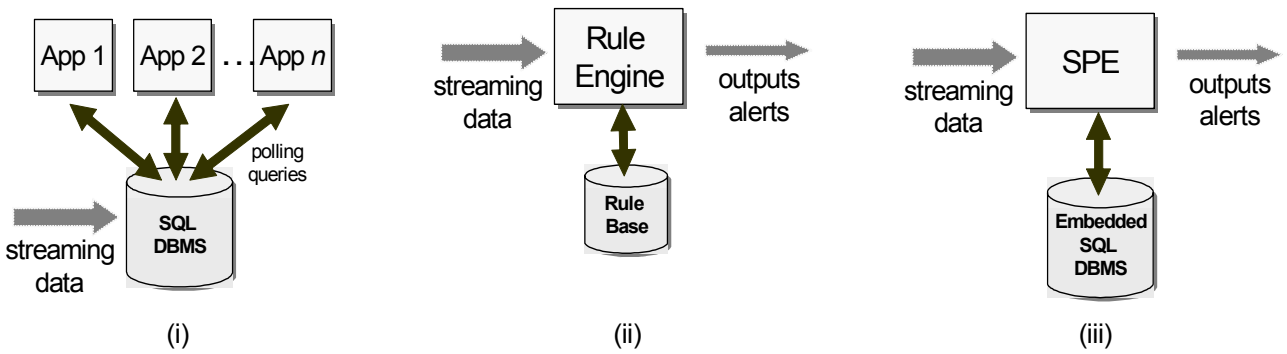


Figure 4: Basic architectures of (i) a database system, (ii) a rule engine, and (iii) a stream processing engine.

code additional ones in a general purpose language such as C++ or Java, using embedded SQL calls to the DBMS.

- **Rule engines** date from the early 1970's when systems such as PLANNER and Conniver were initially proposed by the artificial intelligence community. A later more widespread rule engine was Prolog (1980s), and there have been several more recent examples (e.g., OPS5 [7]). A rule engine typically accepts condition/action pairs, usually expressed using "if-then" notation, watches an input stream for any conditions of interest, and then takes appropriate action. In other words, a rule engine enforces a collection of rules that are stored in a rule base.

  Figure 4(ii) illustrates the basic rule engine model for stream processing. The rule base provides persistent storage for rules. As streaming data enters the system, they are immediately matched against the existing rules. When the condition of a rule is matched, the rule is said to "fire". The corresponding action(s) taken may then produce alerts/outputs to external applications or may simply modify the state of internal variables, which may lead to further rule firings.

- **Stream Processing Engines (SPEs)** are specifically designed to deal with streaming data and have recently gotten attention as a third alternative. Their basic architecture is shown in Figure 4(iii).

  SPEs perform SQL-style processing on the incoming messages as they fly by, without necessarily storing them. Clearly, to store state when necessary, one can use a conventional SQL database embedded in the system for efficiency. SPEs use specialized primitives and constructs (e.g., time-windows) to express stream-oriented processing logic.

Next, we briefly evaluate these systems on the basis of the requirements we presented in Section 2.

3.2 How do they measure up?

DBMSs use a "process-after-store" model, where input data are first stored, potentially indexed, and then get processed. Main-memory DBMSs are faster because they can avoid going to disk for most updates, but otherwise use the same basic model. DBMSs are passive; i.e., they wait to be told what to do by an application. Some have built-in triggering mechanisms, but it is well-known that triggers have poor scalability. On the other hand, rule engines and SPEs are both active and do not require any storage prior to processing. Thus, DBMSs do not **keep the data moving**, whereas rule engines and SPEs do.

SQL was designed to operate on finite-size stored data and thus needs to be extended in order to deal with potentially unbounded streams of time-series data. SQL/Temporal is still in its infancy and, SQL, as implemented by the DBMS vendors, supports only a rudimentary notion of windowed operations (i.e., sort and aggregate). Rule languages need to be extended in a similar manner so that they can express conditions of interest over time. Moreover, rule languages also need the notion of aggregation, a common operation in many streaming applications. Therefore, SPEs support **SQL-style processing on streams**, whereas DBMSs and rule engines do not.

In rule engines and SPEs, it is possible to code operations that might block. Hence, any implementation of these systems should support time-outs. In a DBMS solution, applications have to explicitly specify their polling behavior to simulate the effect of time-outs and receive partial data. On the other hand, a DBMS triggering system has no obvious way to time out. Dealing with out-of-order data exhibits similar challenges for a DBMS. Overall, **handling stream imperfections** is much easier with rules engines and SPEs than with DBMSs.

To **generate predictable outcomes**, an SPE or a rule engine must have a deterministic execution mode that utilizes timestamp order of input messages. DBMSs have particular difficulty with this requirement simply because they are passive—some external system would have to control the order in which messages were stored and processed. In addition, application programs are fundamentally independent and their execution is controlled by an operating system scheduler. Enforcing some order on the execution of application programs is another task that would have to be done by some external software.

Seamlessly **integrating stored and streaming data** is problematic for both DBMSs and rules engines. Storing state is what DBMSs do naturally. As argued earlier, however, DBMSs cannot cope well with streaming data. Even if only used for state storage in a streaming application, a client-server DBMS will be ineffective as it will incur high latency and overhead. As such, a DBMS solution will only be acceptable if the DBMS can be embedded in the application.

In contrast, a rule engine can effectively keep data moving, but has problems when dealing with state storage. The reason is that a rule engine relies on local variables for storage, and there is no easy way to query local variables. To cope with a large amount of

state, one must then somehow graft an embedded DBMS onto a rule engine. In this case, it is necessary to switch from local variables to a DBMS paradigm, an unnatural thing to do. Hence, a rule engine is ill-suited for storing significant amounts of state information. An SPE, on the other hand, should be able to support and seamlessly integrate streaming and stored data by simply switching the scope of a StreamSQL command from a live feed to a stored table.

All three systems can incorporate appropriate mechanisms to **guarantee data safety and availability**. Similarly, there are no fundamental architectural impediments to prevent these systems from **partitioning and scaling applications**.

Finally, all architectures can potentially **process and respond instantaneously**; however, DBMSs are at a big disadvantage here as they do not employ a straight-through processing model.

3.3 Tabular results

In Table 1, we summarize the results of the discussion in this section. Each entry in the table contains one of four values:

- **Yes**: The architecture naturally supports the feature.
- **No**: The architecture does not support the feature.
- **Possible**: The architecture *can* support the feature. One should check with a vendor for compliance.
- **Difficult**: The architecture *can* support the feature, but it is *difficult* due to the non-trivial modifications needed. One should check with the vendor for compliance.

SPEs offer the best capabilities since they are designed and optimized from scratch to address the requirements of stream processing. Both DBMSs and rule engines were originally architected for a different class of applications with different underlying assumptions and requirements. As a result, both systems fundamentally "shoehorn" stream processing into their own model. It is, thus, not surprising to see that they have fundamental limitations for this domain. In particular, neither system has the capability to efficiently and uniformly deal with *both* streaming and stored data.

	DBMS	Rule engine	SPE
Keep the data moving	No	Yes	Yes
SQL on streams	No	No	Yes
Handle stream imperfections	Difficult	Possible	Possible
Predictable outcome	Difficult	Possible	Possible
High availability	Possible	Possible	Possible
Stored and streamed data	No	No	Yes
Distribution and scalability	Possible	Possible	Possible
Instantaneous response	Possible	Possible	Possible

Table 1: The capabilities of various systems software.

4. CONCLUDING REMARKS

There is a large class of existing and newly emerging applications that require sophisticated, real-time processing of high-volume data streams. Although these applications have traditionally been served by "point" solutions through custom coding, system software that specifically target them have also recently started to emerge in the research labs and marketplace.

Based on our experience with a variety of streaming applications, we presented eight rules to characterize the requirements for real-time stream processing. The rules serve to illustrate the necessary features required for any system software that will be used for high-volume low-latency stream processing applications. We also observed that traditional system software fails to meet some of these requirements, justifying the need for and the relative benefits of SPEs.

REFERENCES

[1] Crossbow Technology Inc., 2005. http://www.xbow.com/.

[2] Lojack.com, 2005. http://www.lojack.com/.

[3] D. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. Aurora: A New Model and Architecture for Data Stream Management. VLDB Journal, 2003.

[4] A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, I. Nizhizawa, J. Rosenstein, and J. Widom. STREAM: The Stanford Stream Data Manager. In ACM SIGMOD Conference, June 2003.

[5] A. Arasu, M. Cherniack, E. Galvez, D. Maier, A. Maskey, E. Ryvkina, M. Stonebraker, and R. Tibbetts. Linear Road: A Benchmark for Stream Data Management Systems. In Very Large Data Bases (VLDB) Conference, Toronto, CA, 2004.

[6] J. Barlett, J. Gray, and B. Horst. Fault tolerance in Tandem computer systems. Tandem Computers TR 86.2., 1986.

[7] L. Brownston, R. Farrell, E. Kant, and N. Martin, Programming Expert Systems in OPS5: Addison-Wesley, 1985.

[8] D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. Zdonik. Monitoring Streams: A New Class of Data Management Applications. In proceedings of the 28th International Conference on Very Large Data Bases (VLDB'02), Hong Kong, China, 2002.

[9] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. R. Madden, V. Raman, F. Reiss, and M. A. Shah. TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. In Proc. of the 1st CIDR Conference, Asilomar, CA, 2003.

[10] E. F. Codd. Does your DBMS run by the rules? ComputerWorld, October 21, 1985.

[11] E. F. Codd. Is your DBMS really relational? Computerworld, October 14, 1985.

[12] E. F. Codd. Providing OLAP to User-Analysts: An IT Mandate. Codd and Associates, Technical Report 1993.

[13] J. P. Corrigan. OPRA Traffic Projections for 2005 and 2006. Technical Report, Options Price Reporting Authority, Aug, 2005. http://www.opradata.com/specs/projections_2005_2006.pdf.

Citation analysis of database publications

Erhard Rahm, Andreas Thor
University of Leipzig, Germany
{rahm | thor }@informatik.uni-leipzig.de

Abstract

We analyze citation frequencies for two main database conferences (SIGMOD, VLDB) and three database journals (TODS, VLDB Journal, Sigmod Record) over 10 years. The citation data is obtained by integrating and cleaning data from DBLP and Google Scholar. Our analysis considers different comparative metrics per publication venue, in particular the total and average number of citations as well as the impact factor which has so far only been considered for journals. We also determine the most cited papers, authors, author institutions and their countries.

1. Introduction

The impact of scientific publications is often estimated by the number of citations they receive, i.e. how frequently they are referenced by other publications. Since publications have associated authors, originating institutions and publication venues (e.g. journals, conference proceedings) citations have also been used to compare their scientific impact. For instance, one commonly considered indicator of the quality of a journal is its *impact factor* [AM00]. The impact factors are published yearly by Thomson ISI in the Journal Citation Report (JCR) by counting the citations from articles of thousands of journals.

However, database research results are primarily published in conferences which are not covered by the JCR citation databases. The two major database conferences, SIGMOD and VLDB, receive and publish many more papers than the two major journals, ACM TODS and VLDB Journal (VLDBJ). Furthermore, these conferences are more than twice as selective as the journals with acceptance rates for research papers of 15-20% vs. 35-45% [Be05]. The number of conference submissions has increased significantly in the last five years [Be05] underlining the high scientific importance of conferences.

The tremendous scope of new scientific archives like Google Scholar makes it possible to freely access citation data for millions of publications and authors and thus to evaluate the citations for entire conferences and journals. For our analysis we utilized our new data integration platform iFuice [Ra05] to combine bibliographic data on conferences and journals from DBLP with citation data from Google Scholar and the ACM Digital library. We evaluate citations for all papers which appeared between 1994 and 2003 in the two conferences SIGMOD and VLDB, and the three journals TODS, VLDBJ and Sigmod Record (SR). The latter is not a refereed journal but more a newsletter which also publishes short research articles of broader interest. It has good visibility in the database community favored by its free online accessibility.

In the next section we briefly discuss previous attempts to evaluate the citation impact of database conferences and journals. Section 3 provides information on the data sources and the data cleaning applied. Sections 4-9 present our comparative citation analysis for the five publication venues. In particular, section 6 analyzes the citation skew, section 7 evaluates the journal and conference citation impacts, section 8 discusses the most frequently referenced papers and authors, and section 9 determines the most referenced institutions and their countries.

2. Previous evaluations

The DBLP website contains a list of the 120 most referenced database publications with a total of about 17,000 citations[1]. The list was determined from about 100,000 citations in the SIGMOD anthology containing research papers from 1975-1999. The list contains 17 books, 11 papers from ACM Computing Surveys, 29 TODS papers, 22 SIGMOD, 6 VLDB, only 1 VLDBJ and no SR paper. Most citations go to the classic papers from the seventies and early eighties, the most referenced paper being the 1976 TODS paper by Chen on the entity-relationship model (608 citations). The youngest entry is from 1996 and only 9 publications have appeared after 1990 so that this list does not reflect the citation impact for the more recent research. Furthermore, the list only reflects citations from the database publications of the anthology but not from other publication venues or related fields.

Citeseer is a large archive of computer science publications collected from the web. Based on the citations found in its document base, publication venues (journals, conferences, workshops, newsletters etc.) which received at least 25 citations were ranked according to their average number of citations per referenced paper[2]. In a list of more than 1200 venues TODS and VLDBJ achieved ranks 51 and 52, SIGMOD rank 66, VLDB rank 106 and SR rank 414. There are several problems with this ranking. First of all, not all papers of a venue are considered but only those which have at least one citation in the Citeseer collection. Second, as already observed in [Sn03] Citeseer includes many unreviewed technical reports but lacks many publications from database journals and conferences. Thirdly, the average number of citations favors venues with a smaller number of papers like

[1] http://www.informatik.uni-trier.de/~ley/db/about/top.html
[2] http://citeseer.ist.psu.edu/impact.html

journals or workshops compared to larger conferences. As an example, the WebDB workshop (associated with the SIGMOD conference) achieved rank 35 and is thus ranked higher than SIGMOD itself. Finally, the list was last generated in May 2003 and thus does not reflect more recent publications. In general, Citeseer seems to become quickly outdated since only comparatively few new documents are added.

3. Data sources and data integration

Our study is based on data from three sources as of August 2005: the DBLP bibliography[3], Google Scholar (GS)[4] and the ACM Digital Library (ACMDL)[5]. DBLP and ACMDL provide bibliographic information on authors, publishers and complete lists of papers per conference and journal. ACMDL also provides many conference and journal documents and citation counts. However, only citations from the documents in the ACMDL collection are considered. Google Scholar covers a huge number of documents by crawling the web automatically but also includes the papers from several digital libraries including those from ACMDL, IEEE, and Springer. GS automatically extracts the bibliographic data from the reference sections of the documents (mostly in PDF and PS formats) and determines citation counts for papers in its collections as well as for citations for which the document is not available. The publications in the result of a query are typically ranked according to the number of citations.

We use our integration platform iFuice [Ra05] to combine the data from the mentioned sources and determine the number of citations for entire conferences and journals. We map each paper found in DBLP for a given publication venue to both ACMDL and GS to determine its citations. Moreover, we perform extensive data cleaning to deal with errors in the citations and limitations of the automatic extraction of references. For instance, GS frequently has several entries for the same paper, e.g. due to misspelled author names, different ordering of authors etc. On the other hand, GS may group together citations of different papers, e.g. for a journal and conference version of a paper with the same or similar title. In addition to dealing with these issues we also determine all author self-citations in order to eliminate them from the citation counts. To extend the scope of our analysis we also did some manual data preparation. In particular, we grouped papers into different types (research, industrial, demo, panel, etc.) and determined the originating institution of papers.

Fig. 1 shows the normalized number of citations for GS and ACMDL to all considered papers published between 1994 and 2003. 100% refers to the total number of GS citations including self-citations. The other curves indicate the shares for GS citations without self-citations, the GS citations from publications for which GS has an associated publication year, and the ACMDL citations without self-citations. The graph shows that on average about 10% of the GS citations are self-citations (i.e. about 90% of the GS citations remain) and that this value increases somewhat to about 17% for recent publications. Only for about 50-60% of its citations GS has the year of the referencing paper. This information is needed to determine the impact factor or the age of citations. GS derives the year of a publication X apparently from citations to X so that the year is often unknown for unreferenced papers. This also explains why the share of GS citations with a year information goes down for more recent years.

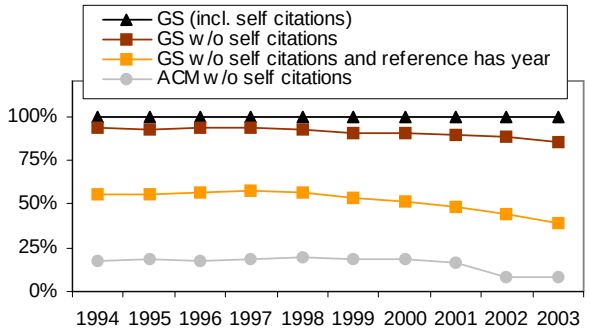


Figure 1: Comparison of different citation counts (100% = Google scholar including self citations)

The number of ACM citations is only about 20% of GS citations until 2001. For 2002 and 2003 the share goes down to about 10% indicating that the ACMDL is less current than GS (in fact, all VLDB papers from 2002 and younger were missing in ACMDL as of August 2005). For these reasons we will only consider the cleaned *GS citations without self-citations* for the remaining analysis. The calculation of impact factors is based on GS citations with a known year for the referencing publication.

4. Base statistics

We determined the number of citations for all papers listed at DBLP for the five considered publication venues and the ten years 1994 - 2003. As of August 2005, we had 81,680 cleaned GS citations for 2,338 papers.

In a first step we analyzed the distribution of citations over different types of papers. Both SIGMOD and VLDB publish not only research papers but also industrial and application papers, demo descriptions, panel and tutorial abstracts and invited papers (mostly with an abstract only). Fig. 2 compares the relative number of papers with the relative citation frequency for these publication types. The "non-research" papers account for a substantial share of the publications, namely 34% and 41% for VLDB and SIGMOD, respectively. However, they receive less than 10% of the citations for both conference series and thus have only a limited impact. This is probably because many of these pa-

[3] http://www.informatik.uni-trier.de/~ley/db/index.html
[4] http://scholar.google.com
[5] http://portal.acm.org

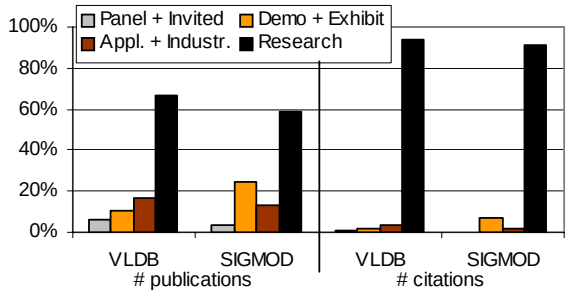


Figure 2: Relative impact of conference paper type

Conference / Journal	# Publications	# Citations	avg. # citations per publication
SIGMOD	446	31,069	70
VLDB	570	28,659	50
SIGMOD Record	327	7,724	24
VLDB Journal	189	4,919	26
TODS	130	4,162	32
Overall	1,662	76,533	46

Table 1: Number of publications and citations

pers are very short (1-4 pages) and the acceptance process is less selective than for research papers. In our remaining analysis we will therefore focus on research papers. For this purpose, we also exclude about 20% of the SR articles like editorials, book reviews, interviews, obituaries etc. from further consideration.

Table 1 summarizes the total number of publications and citations and the average number of citations received per paper for the five publication venues. Most papers (ca. 61%) are published in the two conference series. What is more the total number of citations for SIGMOD and VLDB is almost by a factor 7 higher than for TODS and VLDBJ. Hence the journals have only a comparatively small citation impact. Even with respect to the average number of citations the conferences are clearly ahead by more than a factor 2. This is likely because the most up-to-date research results are primarily published in conferences. Successful conference submissions are published within six to seven months while the so-called end-to-end time for journals (time delay between submission of a manuscript and publication time of the issue with the article) is much higher and highly variable. As outlined in [Be05] the average end-to-end time used to be 2 to 3.3 years for TODS and 1.5 to 2.8 years for VLDBJ in the nineties; currently both journals have an improved average of about 1.5 years.

SIGMOD published fewer papers than VLDB but received more citations resulting in a 40% higher average number of citations per paper. TODS and VLDB Journal are closer together. TODS achieved a better average number of citations while VLDB Journal received more citations in total. SR publishes substantially more (short) papers than either TODS or VLDBJ and achieves a surprisingly high number of citations. As we outline below this is mainly because of some heavily referenced and timely survey papers.

5. Development over time

We now drill down from the summary data into the time dimension to see the distribution over the 10 years. Figures 3-6 show for each publication venue and year the number of papers, the total number of citations, the number of citations to the 5 most referenced papers and the average number of citations per paper.

Fig. 3 indicates that TODS and VLDBJ have a relatively constant and low number of publications per year. (The atypical VLDBJ numbers for 1999 and 2000 are due to a delayed issue.) VLDB publishes most papers per year and recently increased the number of research papers from around 50 to 75 in 2003 to keep pace with an increased number of submissions.

Fig. 4 illustrates the much higher number of citations for the two conferences compared to the journals. Most citations refer to papers from the nineties while the number of citations continuously decreases since 1999. This indicates that many references reach back five and more years giving younger papers comparatively little opportunity to get referenced. The most referenced conferences achieved almost 5000 citations (VLDB94, SIGMOD96, SIGMOD98) while the more recent ones have earned less than 2000 citations so far. Despite the higher number of papers VLDB is referenced more than SIGMOD only in 4 of the 10 years (1994, 1995, 2001, 2002).

Comparing Fig. 4 with Fig. 5 illustrates that the 5 most referenced papers already account for a large portion of all citations. In fact, they receive on average about 40% of all citations for the conferences and 70% for the journals. Interestingly, the top-5 referenced conference papers are on average three times as frequently cited than the top-referenced journal papers. In a few cases (SR1997, SR1998, VLDBJ2001) survey articles helped the journals to close the gap to the conferences.

Fig. 6 shows that SIGMOD dominates w.r.t. the average number of citations especially for the four years 1995-1998 with an average of about 100 references per paper. Another observation is that in the last 5 years the averages for journals and conferences have approached each other.

6. Citation skew per venue

Metrics for entire publication venues like impact factors and total / average number of citations do not allow one to estimate the impact of individual papers or authors. This is because the distribution of citations is typically highly skewed across different papers as already seen from the impact for the 5 most referenced papers (Fig. 5). We now analyze the degree of citation skew in somewhat more detail for the five publication venues.

In a first approach we sort the papers per venue w.r.t. to their citation count and group them into 4 quarters with the

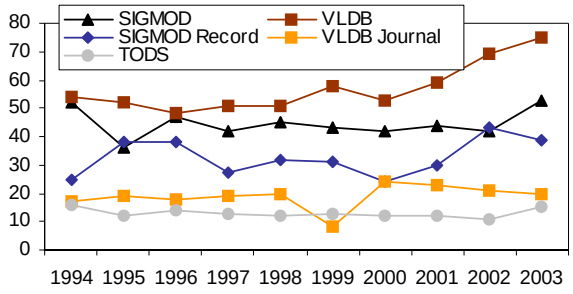


Figure 3: Number of publications

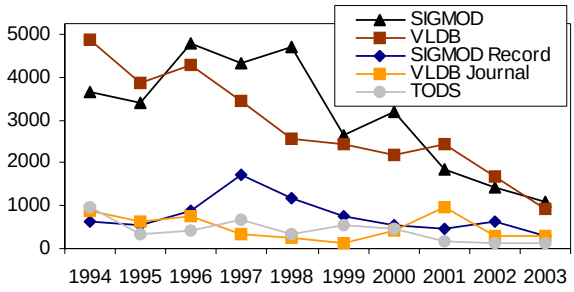


Figure 4: Total numer of citations

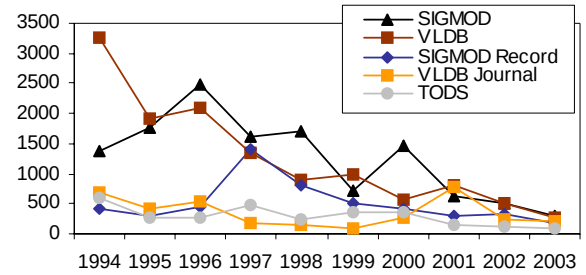


Figure 5: Number of citations for top 5 publications

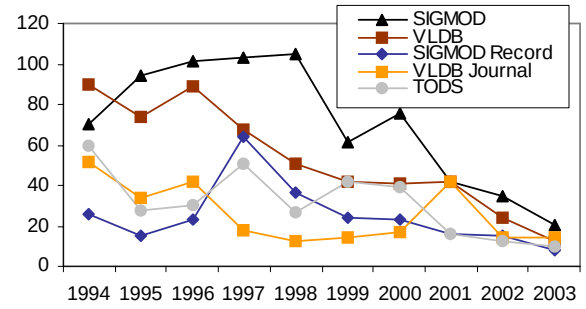


Figure 6: Average number of citations

same number of papers. For each quarter we then determine the relative cumulative citation count. As indicated in Fig. 7 the 25% top-referenced papers account for 60 to 80% of all citations while the bottom 25% of the papers merely achieve 2 - 5% of all citations. In this regard, SR exhibits the highest skew. By contrast, TODS is the most balanced publication venue. VLDB is more skewed than SIGMOD, i.e., the most referenced publications dominate the overall number of citations more and the least referenced papers have even less citation impact for VLDB than for SIGMOD. This might be influenced by the larger number of papers at VLDB compared to SIGMOD. Interestingly, VLDB and VLDB Journal have a very similar citation distribution.

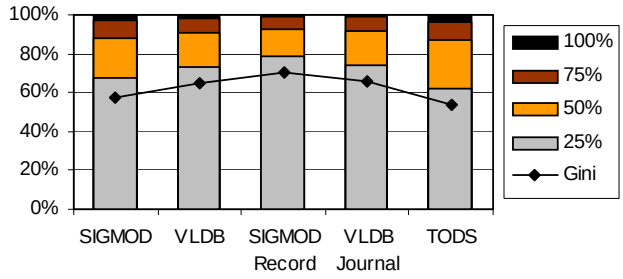


Figure 7: Citation distribution (splitted by quarters) and Gini inde

As a second, handier metric for the citation skew we consider the so-called Gini coefficient using the Brown formula[6]. The Gini index is a measure of (in our case: citation) inequality. It is a number between 0 and 1 where 0 corresponds to perfect equality (i.e., all publications have the same number of citations) and 1 corresponds to complete inequality (i.e., one paper has all citations). As indicated in Fig. 7 the Gini coefficients confirm our observations above with the highest value for Sigmod Record (0.7) and the lowest for TODS (0.53).

7. Impact factor

The journal impact factor (JIF) determines the average number of citations per paper for a period of two years. For a given year X, the JIF is the average number of times articles from the journal published in the past two years X-2 and X-1 have been cited in year X. For example, the JIF for VLDB Journal in the year 2003 is calculated by dividing the number of 2003 citations to VLDBJ papers from 2001 and 2002 by the number of VLDBJ papers in 2001 and 2002. For the citations recorded in the Journal Citation Report (JCR) database the result is

$$JIF_{VLDBJ2003} = (148 + 52) / (23 + 21) = 4.545$$

which made VLDBJ one of the top-rated computer science journals in 2003. Fig. 8 shows the available JIF values from JCR for the three database journals. The curves indicate that all journals have increased their impact factors in the last two years (which might be influenced by an increased number of evaluated papers). VLDBJ has seen the largest increase thereby outperforming TODS which in turn outperforms SR[7]. The JCR contains additional metrics like the number of citations within a year to all previous articles of a journal (not

[6] http://en.wikipedia.org/wiki/Gini_coefficient

[7] The JCR impact factors for 2002 and 2003 are likely flawed (too low) for SR because they are based on a too high a number of papers (105 and 109 papers for 2000 and 2001 compared to 47 for 2002 and 2003). This underlines the importance of data quality (data cleaning) for citation analysis

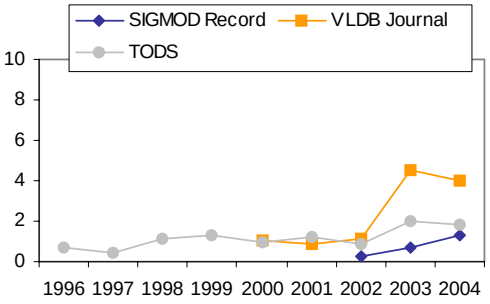


Figure 8: JCR impact factors for journals (2 years)

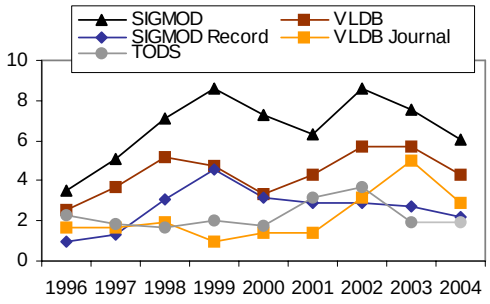


Figure 9: GS-based impact factors (2 years)

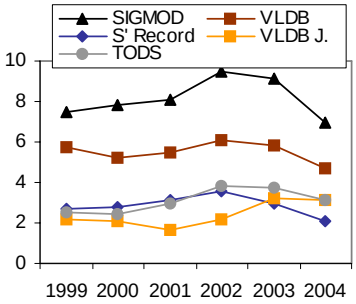


Figure 10: GS-based impact factors (5 years)

only to articles of the two previous years) and the so-called half-life, i.e. the number of recent years accounting for 50% of all references. For instance, there are 870 TODS citations vs. 755 VLDBJ citations in 2004. However, the half-life of TODS (launched in 1976) is more than 10 years, i.e. most citations refer to older articles, which is in line with our observations from the DBLP ranking (section 2). On the other hand, VLDBJ was launched in 1992 and has a half-life of only 4 years, i.e. 50% of the 2004 citations refer to articles from 2001 and younger.

We used the GS citations with a known year (section 3) to determine the impact factors not only for the journals but also for the conferences. In addition, we not only calculated the impact factors over two but also over five years (for instance the five-year impact factor for 2004 indicates the average number of citations in 2004 publications to publications from 1999-2003). The consideration of more than two years was proposed in [Am00] to improve the impact factors' coverage and reduce fluctuations due to a few highly cited articles.

Fig. 9 and Fig. 10 show the resulting impact factors for the five publication venues. We see that the impact factors for the journals are mostly higher than in Fig. 8 because GS provides more citations than the JCR database (despite the elimination of self-citations and citations for which the year of the citing publication is unknown). Furthermore, the impact factor is more stable than the total and average number of citations of Figs. 4 and 6 since the uniform window of 2 and 5 years reduces the bias against younger papers which have less time than older papers to get referenced. The most striking result is that the two conferences achieve excellent impact factors and outperform the journals in all years. SIGMOD achieves the best impact factor in all years. As for the two-year JCR impact factor, VLDBJ reaches the best impact factor of the three journals in the last two years (Fig. 9). SR achieves a surprisingly good impact factor favored by our elimination of non-research papers during data preparation. The five-year impact factors (Fig. 10) are less influenced by a few heavily cited papers. For this extended evaluation period, all journals remain clearly behind the two conferences in all years.

8. Most referenced papers and authors

Tables 2 and 3 show the 5 most cited conference and journal publications, respectively. Longer lists and the top 5 papers per publication venue and year can be found online at *www.ifuice.de*. The by far most cited publication in the considered time frame is Agrawal's and Srikant's 1994 association rule paper (which received the 10-Year-Best-Paper-Award at VLDB 2004). Data mining papers actually contribute substantially to the high citation counts of the conferences in the nineties: 10 from the 20 most referenced papers belong to this category. Their high impact is attributable to

	Title	Authors	Published in	#Cit.
1.	Fast Algorithms for Mining Association Rules	R. Agrawal, R. Srikant	VLDB '94	2261
2.	Querying Heterogeneous Information Sources Using Source Descriptions	A.Y. Levy, A. Rajaraman, J.J. Ordille	VLDB '96	692
3.	BIRCH: An Efficient Data Clustering Method for Very Large Databases	T. Zhang, R. Ramakrishnan, M. Livny	SIGMOD '96	617
4.	Mining Frequent Patterns without Candidate Generation	J. Han, J. Pei, Y. Yin	SIGMOD '00	573
5.	Implementing Data Cubes Efficiently	V. Harinarayan, A. Rajaraman, J.D. Ullman	SIGMOD '96	559

Table 2: Most referenced conference publications (1994-2003)

	Title	Authors	Published in	#Cit.
1.	An Overview of Data Warehousing and OLAP Technology	S. Chaudhuri, U. Dayal	SR '97	634
2.	Lore: A Database Management System for Semistructured Data	J. McHugh, S.Abiteboul, R.Goldman, D.Quass, J.Widom	SR '97	392
3.	Database Techniques for the World-Wide Web: A Survey	D. Florescu, A.Y. Levy, A. Mendelzon	SR '98	391
4.	A Survey of Approaches to Automatic Schema Matching	E. Rahm, P.A. Bernstein	VLDB J '01	324
5.	An Introduction to Spatial Database Systems	R.H. Güting	VLDB J '94	280

Table 3: Most referenced journal publications (1994-2003)

	Author	# Cit.	# Pub.
1.	Rakesh Agrawal	5393	21
2.	Ramakrishnan Srikant	3654	7
3.	Alon Y. Halevy	3052	25
4.	Hector Garcia-Molina	2792	47
5.	Jeffrey F. Naughton	2657	34
6.	Michael J. Franklin	2475	26
7.	David J. DeWitt	2328	27
8.	Jennifer Widom	2176	22
9.	Jiawei Han	1997	17
10.	Christos Faloutsos	1960	22

Table 4: Most referenced authors

the fact that they successfully reached out of the database field to other communities as well. Surprisingly, journal papers on data mining did not appear in the database journals but apparently elsewhere. As Table 3 indicates 4 from the 5 most referenced journal papers are surveys on topics with a substantial research activity to follow on. These surveys helped SR to achieve good citation numbers in 1997/1998 and VLDBJ in 2001, and good two-year impact factors 2 years later.

Table 4 lists the 10 most cited authors together with the number of their papers in the considered venues and time frame. In case of several co-authors per paper, we attributed all citations to each co-author. Four of the ten authors are also in the 2002 list of the ten smallest centrality scores indicating a large co-authorship / social network [Na03].

9. Citation counts by country and institution

For our final evaluation we study the distribution of citations over originating institutions and their countries. For simplicity, we only consider the first author's institution which we extracted manually from the papers. This is obviously very time-consuming so that we restricted ourselves to the research papers with at least 20 citations. Overall, we considered the top-referenced 50% research publications receiving more than 91% of all citations so that we believe that the results are still significant.

Tables 5 and 6 list the top 10 countries and institutions, respectively, with respect to these citation counts. Note that the absolute values cannot directly be compared with the previously presented numbers due to the reduced set of papers. Moreover we attribute the citations of a paper only to one country and one institution, while for Table 4 the citations were assigned to each co-author.

Table 5 shows that almost three quarter of all citations go to papers from US institutions which also contribute by far the most papers. Runners-up are Germany and Canada but with a huge difference to the US. Papers from France and Italy still achieve a noticeable citation impact while countries like UK do not make it on the list.

Table 6 shows that IBM and Stanford are the institutions with the highest citation impact. Only two non-US institutions are listed, the French institute INRIA and the German University of Munich. Comparing Table 5 with Table 6 indicates that papers from some institutions receive more citations than entire countries other than the USA. For example, papers from all German universities combined receive fewer citations than Stanford university alone.

10. Conclusions

We presented a detailed comparative citation analysis for five database publication venues. We note that the two main database conferences SIGMOD and VLDB have a substantially higher citation impact than the database journals TODS, VLDBJ and Sigmod Record, not only in terms of the total number of citations but also with respect to the two-year and five-year citation impact. SIGMOD achieves a higher citation impact than VLDB. Sigmod Record and VLDBJ have benefited from survey articles, while TODS has the least skewed distribution of citations. US institutions receive almost 75% of all citations with papers from IBM and Stanford receiving most citations. The study underlines the high usefulness of Google Scholar for evaluations like this. Data preparation, data cleaning and integrating data from several sources are important to achieve useful and correct results showing the value of tools like iFuice.

Acknowledgements We thank Phil Bernstein, Tamer Özsu, and Rick Snodgrass for their insightful comments.

	Country	# Cit.	in %	# Pub.
1.	USA	51222	73.2	567
2.	Germany	4291	6.1	64
3.	Canada	3270	4.7	34
4.	France	2222	3.2	29
5.	Italy	2025	2.9	22
6.	Israel	858	1.2	6
7.	Japan	724	1.0	6
8.	Denmark	644	0.9	7
9.	Switzerland	612	0.9	8
10.	Greece	590	0.8	12

Table 5: Citations by country

	Institution	# Cit.	# Pub.
1.	IBM	9540	70
2.	Stanford University	7045	62
3.	University of Wisconsin-Madison	5132	60
4.	Bell Labs & AT&T Labs	4500	55
5.	University of Maryland	3153	32
6.	Microsoft	2360	24
7.	University of California, Berkely	1908	24
8.	INRIA (France)	1854	20
9.	University of Washington	1487	15
10.	University of Munich (Germany)	1342	13

Table 6: Citations by institution

11. References

AM00 Amin, M., Mabe, M.: Impact Factors: Use and Abuse. Perspectives in Publishing, Oct. 2000

Be05 Bernstein, P.A. et.al.: Database Publication Practices. Proc. 31st VLDB Conf., 2005

Na03 Nascimento, et al.: Analysis of SIGMOD's co-authorship graph. SIGMOD Record 32(3), 2003

Ra05 Rahm, E., Thor, A., et al.: iFuice - Information Fusion utilizing Instance Correspondences and Peer Mappings, Proc. 8th WebDB, 2005

Sn03 Snodgrass, R.: Journal Relevance. SIGMOD Record 32(3), 2003

A Citation-Based System to Assist Prize Awarding

Antonis Sidiropoulos
Data Engineering Lab, Department of Informatics,
Aristotle University, Thessaloniki, 54124 Greece
asidirop@csd.auth.gr

Yannis Manolopoulos
Data Engineering Lab, Department of Informatics,
Aristotle University, Thessaloniki, 54124 Greece
manolopo@csd.auth.gr

Abstract

Citation analysis is performed to evaluate the impact of scientific collections (journals and conferences), publications and scholar authors. In this paper we investigate alternative methods to provide a generalized approach to rank scientific publications. We use the SCEAS system [12] as a base platform to introduce new methods that can be used for ranking scientific publications. Moreover, we tune our approach along the reasoning of the prizes '*VLDB 10 Year Award*' and '*SIGMOD Test of Time Award*', which have been awarded in the course of the top two database conferences. Our approach can be used to objectively suggest the publications and the respective authors the are more likely to be awarded in the near future at these conferences.

1. Introduction

Citation analysis is performed to evaluate the impact of scientific collections (journals and conferences), publications and scholar authors during hiring, promotion, tenure and merit pay decisions. The first study in this field appeared in 1972 [4], however we meet many similar studies in the literature up to the present [10, 11].

In general, the ranking algorithms that are used in bibliometrics can be separated into two classes. We call the first one *collection-based ranking algorithms*. At this class of algorithms, a weighted citation graph is used, where the collections are the graph nodes, whereas the weighted edges represent the total number of the citations that point from one collection to another. The ISI Impact Factor belongs to this ranking class [3, 4, 5].

Our alternative approach, the SCEAS (Scientific Collection Evaluator with Advanced Scoring) method and system which has been presented in [12], also belongs in the same ranking class. SCEAS is a web-based digital library and its contents are imported from the Data Bases and Logic Programming (DBLP) website[1] of the University of Trier. The SCEAS system emerges as a useful tool for conducting several experiments on the DBLP data. Most of the results that are presented in this paper (as well as other results) are accessible through the SCEAS website[2].

We call the second class of ranking methods *publication-based ranking algorithms*. According to this approach, the nodes of the citation graph represent publications, whereas an edge from node x to node y represents a citation from paper x to paper y. The advantage of ranking at the publication level is that it is possible to evaluate more than one entity through a single computation: the paper itself, the collection where it belongs, and the author(s) as well. The last two evaluations can be made by an aggregate average of the first one. All the ranking algorithms that where initially proposed to rank web pages can be categorized in this class. In this respect, two of the most well known algorithms are the PageRank by Brin and Page [1] and HITS by Kleinberg's, which classifies the graph nodes as Hubs and Authorities [7, 8, 9].

The structure of this paper is as follows. In the next section, we briefly present the ranking methods for the *publication-based ranking* (PageRank and HITS) and we criticize their weakness when used in bibliometrics. We also investigate new alternative methods to overcome the vulnerabilities of the above algorithms, with respect to our specific goal. These new methods have been used within the SCEAS system, and thus we named our family of ranking algorithm as *SCEAS Rank*. Section 3 shows the algorithm performance with respect to the computation speed. Section 4 is divided in two parts, each part containing the rank results over the DBLP data for publications and authors respectively. In this section, we also evaluate the potential of the ranking methods by tuning and comparing their results to the well known awards '*VLDB 10 Year Award*' and '*SIGMOD Test of Time Award*', which have been given in the recent past in the two most prominent conferences of the database community.

2. Ranking Algorithms

In this section we introduce the well known PageRank and HITS algorithms and we investigate the reasons of their deficiency when they are used for publications ranking. We also present two new ranking methods.

2.1 HITS

HITS algorithm has been proposed as a method for ranking web pages that are retrieved when searching via a browser. It is based on two specific notions: *hubs* and *authorities*. In particular, the score of hubs and authorities is calculated by using the equations:

$$\begin{aligned} \vec{a'} &= A^T \vec{h} \\ \vec{h'} &= A * \vec{a} \end{aligned} \tag{1}$$

where $\vec{a}$ and $\vec{h}$ are vectors containing the scores of the publications as authorities and hubs respectively. A is the adjacency matrix of the citation graph.

Despite the popularity of HITS in the web context, HITS is not quite suitable in the field of bibliometrics. The reason is that a publication gets high authority score iff there are hubs pointing to it. However, this should not be the main metric in bibliometrics. Kleinberg proposed a model where authorities directly endorse other authorities in [7]. This concept it termed *Prestige* by Chakrabarti in [2]. However, after a closer look, it can be concluded that this approach seems to have problems when the graph does not include any cycles. In this case, the scores of the nodes converge to zero.

2.2 PageRank

PageRank is also popular in the web context. PageRank uses the following formula to calculate the score S_j of an object j (a page

[1] http://www.informatik.uni-trier.de/~ley/db/

[2] http://delab.csd.auth.gr/sceas

in the web context, or a publication in the context of bibliometrics):

$$S_j = (1-d) + d * \sum_{i \to j} \frac{S_i}{N_i} \quad (2)$$

where N_i is the number of citations from publication i, and d is a *damping factor*, usually set to 0.85.

PageRank associates high ranking to node i, if there is a big connected component C where some of its nodes point to i. The more and larger cycles C contains and i belongs to, the bigger score i will come up with. However, cycles do not usually exist in bibliometrics, but if they do so, they are usually self-citations. Thus, it is not fair for the self-citations to affect the score positively. On the other hand, removing the self-citations or the cycles will invalidate the graph and the results. Thus, PageRank could not be fair. This is verified by the graph instance with 12 nodes of Figure 1 and the results of Table 1, where we depict the ranking score for each node by applying all the ranking methods. Table 1 consists of five columns. Each column presents the ranking by the appropriate method and includes 2 sub-columns. Sub-column 1 is the *node id* and sub-column 2 is the *node score* rounded to 2 decimal digits. In this example, node 0 gets 4 citations, while nodes 10 and 6 get 3 citations each. However, the PageRank score of nodes 10 and 6 is about 3 times higher than the score of node 0. This happens because nodes 10 and 6 are parts of citation cycles.

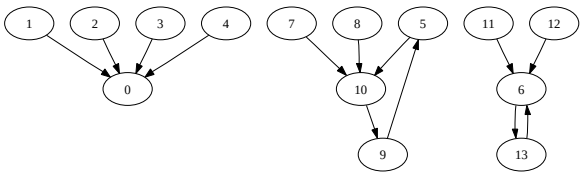


Figure 1. Example 1.

HITS_AU		PageRank		SCEAS1		SCEAS2		Citations	
0	1.00	6	1.92	0	1.47	0	0.34	0	4
1	0.00	13	1.78	6	1.43	6	0.32	6	3
2	0.00	10	1.66	10	1.36	10	0.32	10	3
3	0.00	9	1.56	13	0.90	13	0.25	5	1
4	0.00	5	1.48	9	0.87	9	0.25	9	1
5	0.00	0	0.66	5	0.69	5	0.23	13	1
6	0.00	1	0.15	1	0.00	1	0.15	1	0
7	0.00	2	0.15	2	0.00	2	0.15	2	0
8	0.00	3	0.15	3	0.00	3	0.15	3	0
9	0.00	4	0.15	4	0.00	4	0.15	4	0
10	0.00	7	0.15	7	0.00	7	0.15	7	0
11	0.00	8	0.15	8	0.00	8	0.15	8	0
12	0.00	11	0.15	11	0.00	11	0.15	11	0
13	0.00	12	0.15	12	0.00	12	0.15	12	0

Table 1. Rank tables for citation graph in Figure 1.

Secondly, PageRank is designed in a way (which is suitable for the web) that a page score is mostly affected by the scores of the web pages that point to it and less by the number of the incoming links (citations). A case like this is shown in Figure 2 and Table 2, where node 0 gets higher score than node 1, although node 1 gets 6 citations.

In addition to the above cases, any change of node j score affects the score of node i even if the path connecting them is very long. Neither is required in bibliometrics. In other words, in cases where direct citations exist, it is necessary for the impact to be (a) large and (b) much less when the distance between the two nodes gets larger. This is depicted in Figure 3. Adding a link to node 6 in Figure 4 results in a 7.14% increase of node 5 score and 6.82% of node 4 score, which are quite distant. This is yet another case where PageRank has not been proved to have good behavior in bibliometrics.

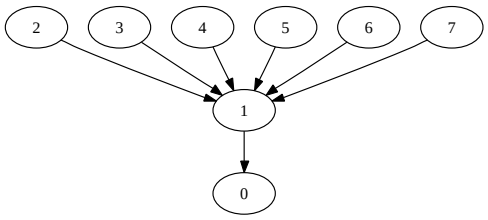


Figure 2. Example 2.

HITS_AU		PageRank		SCEAS1		SCEAS2		Citations	
1	1.00	0	0.93	1	2.21	1	0.43	1	7
0	0.00	1	0.92	0	1.18	0	0.29	0	1
2	0.00	2	0.15	2	0.00	2	0.15	2	0
3	0.00	3	0.15	3	0.00	3	0.15	3	0
4	0.00	4	0.15	4	0.00	4	0.15	4	0
5	0.00	5	0.15	5	0.00	5	0.15	5	0
6	0.00	6	0.15	6	0.00	6	0.15	6	0
7	0.00	7	0.15	7	0.00	7	0.15	7	0

Table 2. Rank tables for citation graph in Figure 2.

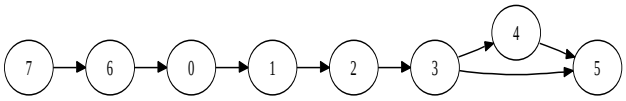


Figure 3. Example 3.

HITS_AU		PageRank		SCEAS1		SCEAS2		Citations	
5	0.85	5	0.77	5	0.76	5	0.24	5	2
4	0.53	3	0.62	3	0.58	3	0.22	0	1
0	0.00	2	0.56	2	0.57	2	0.22	1	1
1	0.00	1	0.48	1	0.55	1	0.22	2	1
2	0.00	4	0.41	0	0.50	0	0.21	3	1
3	0.00	0	0.39	6	0.37	6	0.20	4	1
6	0.00	6	0.28	4	0.29	4	0.18	6	1
7	0.00	7	0.15	7	0.00	7	0.15	7	0

Table 3. Rank tables for citation graph in Figure 3.

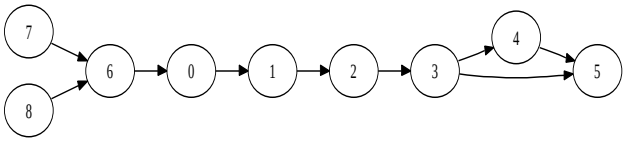


Figure 4. Figure 4.

HITS_AU		PageRank		SCEAS1		SCEAS2		Citations	
5	0.85	5	0.82	5	0.77	6	0.24	5	2
4	0.53	3	0.69	6	0.74	5	0.24	6	2
6	0.00	2	0.63	0	0.64	0	0.23	0	1
0	0.00	1	0.57	1	0.60	1	0.22	1	1
1	0.00	0	0.49	2	0.59	2	0.22	2	1
2	0.00	4	0.44	3	0.58	3	0.22	3	1
3	0.00	6	0.41	4	0.29	4	0.18	4	1
7	0.00	7	0.15	7	0.00	7	0.15	7	0
8	0.00	8	0.15	8	0.00	8	0.15	8	0

Table 4. Rank tables for citation graph in Figure 4.

2.3 SCEAS Rank

In this section, under the generic name SCEAS Rank, we propose two new ranking techniques, named SCEAS1 and SCEAS2. All the requirements mentioned in the previous section can be embedded

in a ranking score, S_j for node j, as follows:

$$S_j = \sum_{i \to j} \frac{S_i + b}{N_i} a^{-1} \quad (a \geq 1, b > 0) \tag{3}$$

where N_i denotes the number of outgoing citations of node i, b is the *direct citation enforcement factor* and factor a denotes the speed in which an indirect citation enforcement converges to zero. The above formula depicts the fact that a change of node i score, affects the score of node j that is x nodes away, by the factor a^{-x}. For our tests we have selected a to be equal to e. Therefore, the involvement of node i to the score of node j is less as the distance x between them gets longer. Actually, it converges to zero for $x > 7$.

In Equation 3 we have used the factor b since citations coming from zero scored nodes should also contribute in the score of their citing publications. The value of b may vary, according to the level that the number of the incoming citations should affect the calculated score. In any case, it must be greater than 0, otherwise the scores will converge to zero. In our tests we have used $b = 1$ as the number of the incoming citations should count significantly.

As shown in the tables above for all the examples, the results of SCEAS1 are satisfactory enough. In Table 1, node 0 is ranked first as it gets 4 citations. Nodes 6 and 10 are ranked second and third, respectively. Node 6, like node 10, takes part in citation cycles. As a result their scores are affected by themselves. Fortunately, the score increment that they gain due to cycles is not that high to advance them to the first position. Thus, in this example, SCEAS1 gives better results than PageRank. In Table 2, SCEAS1 puts node 1 in the first place in contrast to PageRank that ranks node 0 in the first place. In the examples of Table 3 and 4, after node 6 is cited by the just added node 8, SCEAS1 places node 6 in the second place. Also, the node 5 score increment is only 1% relatively to its previous one. On the other hand the change of the score of node 6 is 100% (doubled score), since the number of citations that point to it has been doubled. This small example shows clearly that when new nodes and citations are added to the citation graph, the score of the nodes that are far away from the new nodes are not affected significantly. This means that a very fast incremental computation is feasible.

In summary, the advantages of Equation 3 over PageRank and HITS are:

- The score of a node is mainly affected by the number of incoming citations.
- Computation and convergence is very fast (the results are shown in Section 3).
- The score of a node is less affected by the score of distant nodes. This has also the effect that when new nodes and citations are added, the new computation of the score can be incremental by using the previous score vector as initial vector. The new node actually influences the scores of nodes at most 7 nodes away. That's because practically e^{-7} is almost 0 compared with the scores of nodes. So, the incremental computation is very fast, since the new node influences only a small part of the overall graph.

A dumping factor d may be added to Equation 3 without any major effect in the rank results. This would lead to the following equation, which is a generalized formula of SCEAS1 and PageRank:

$$S_j = (1 - d) + d * \sum_{i \to j} \frac{S_i + b}{N_i} a^{-1} \quad (a \geq 1) \tag{4}$$

For $d = 1$ Equation 4 is equivalent to Equation 3. For $b = 0$ and $a = 1$ Equation 4 is equivalent to PageRank. PageRank uses the value of $d = 0.85$, since this value balances the precision and the convergence speed. A value d closer to 1, means better precision for the scores. Also, a value $d = 1$ should lead PageRank to converge to zero. Therefore, a value $d < 1$ is necessary for PageRank. In our generalized formula, it is safe to use any value for d ($0 < d \leq 1$) if $b > 0$. Also, the convergence speed is mainly affected by the factor a rather than d. In our case, it is safe to use a greater factor d, such as 0.99.

In the examples of Tables 1, 2, 3 and 4 the column for SCEAS1 assumes that $d = 1, b = 1$ and $a = e$, whereas SCEAS2 assumes that $d = 0.85, b = 0$ and $a = e$. A rank with $d = 0.85, b = 1$ and $a = e$ is not included in the tests, since this gives equivalent results to SCEAS1 for the dataset of DBLP digital library.

3. SCEAS Rank Speed

According to the definition of SCEAS Rank, it is obvious that we come up with a very fast convergence by using $b = 1$ and $a = e$. In Figure 5, the x-axis represents the number of the iterations that are needed by each algorithm to compute the ranks for the DBLP digital library. Axis y shows the value of

$$\delta = ||\vec{x}_l - \vec{x}_{l-1}||_1 \tag{5}$$

where $\vec{x}_l$ is the vector with the scores $\{S_1, S_2, ...S_V\}$ after l iterations. The termination condition for each algorithm is: $\delta < \epsilon$, where ϵ is a very small number. Actually, as described in [6], this number could not be predefined for PageRank since it depends on the citation graph. It is obvious that each algorithm needs a different value of ϵ as a termination condition. Regardless of this, the plot shows that the lines of SCEAS Rank are much steeper than the other algorithm lines. This means that it converges faster than the other methods no matter what the actual values of δ and ϵ are.

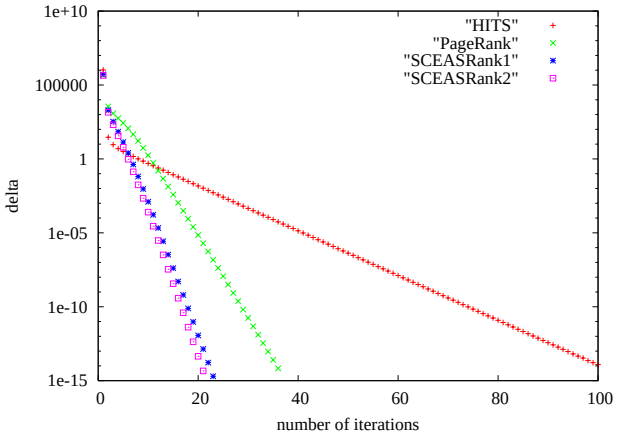


Figure 5. Convergence speeds for DBLP.

Note here that HITS is even slower than what is illustrated in Figure 5. Each iteration of HITS requires about twice the time needed for one iteration of PageRank or SCEAS. This happens because there are two vector multiplications plus a normalization step during each iteration of HITS. The line of HITS should be shifted upwards (as multiplied by 2 at least) to be comparable to SCEAS and PageRank. Conclusively, we can say that SCEAS needs about half the time needed by PageRank and about 1/10 of the time needed by HITS with respect to the DBLP digital library.

4. Results

In this section we present some results on publication and author ranking using data form the DBLP digital Library and our SCEAS

system. Here, it is worth noticing that the DBLP data set is incomplete. The graph out-degrees (citations) are available only for a small part of the total number of the publications contained in the database.

In particular, SCEAS uses the DBLP data included in the inproceedings (370790) and in the articles (217950) collections, which makes a total of 588865 publications as of when our experimentation took place (latest timestamp in DBLP records is 24th Jan 2005). Only 8183 of them (i.e., 1.3%) have their citations actually stored (publications with out-degree). These papers are spanning the time window 1970-2000. The total number of publications during this period is 376912, which means that the citations input is still small. In addition, 18273 publications have in-degree, and thus actually these publications are being ranked.

The total number of citations is 167999 (i.e., the 8183 publications have an average out-degree equal to 20.5). Also, 100210 (i.e., 59.65%) of the above citations point to a publication into the DBLP digital library. The remaining 67789 citations (i.e., 40.35%) point to papers outside the DBLP digital library, and thus they are ignored from further consideration.

Journal	#citations	#cit. in DBLP		#papers	period
ACM Computing Surveys	4471	2733	61%	68	1976-1998
ACM SIGMOD Digital Review	196	185	94%	146	1999-2001
ACM Trans. Database Systems	13313	8011	60%	497	1976-1999
Communications ACM	501	302	60%	31	1970-1999
IEEE Data Eng. Bullettin	2545	1450	57%	178	1991-1999
IEEE Trans. Knowl. Data Eng.	19355	10552	55%	667	1989-1998
SIGMOD Record	1817	1062	58%	99	1981-2000
VLDB Journal	5112	3380	66%	132	1992-2000
journal Summary	47310	27675	58%	1818	1970-2001

Table 5. Journals with citations (out-degree) in DBLP.

Conference	#citations	#cit. in DBLP		#papers	period
ACM SIGMOD	19129	11763	61%	1083	1975-2000
ADBIS	3803	1788	47%	233	1994-1999
CIKM	980	601	61%	55	1995-1995
CoopIS	538	238	44%	30	2000-2000
DASFAA	5042	3131	62%	307	1989-1999
DBPL	3506	2303	66%	148	1987-1997
Digital Libraries	559	170	30%	33	1997-1997
DOLAP	360	223	62%	27	1998-1999
EDBT	4589	2953	64%	244	1988-2000
ER	13267	7114	54%	664	1979-1999
ICDE	18552	11799	64%	1064	1984-1999
ICDT	5239	3593	69%	235	1986-2001
MFDBS	1319	844	64%	68	1987-1991
PDIS	463	307	66%	31	1994-1994
PODS	13081	8580	66%	606	1982-2000
POPL	73	47	64%	4	1979-1982
RIDE	219	113	52%	13	2001-2001
SSDBM	3902	1792	46%	230	1981-1999
VLDB	25887	15059	58%	1277	1975-2000
WIDM	181	117	65%	13	1999-1999
Conferences Summary	120689	72535	60%	6365	1975-2001

Table 6. Conferences with citations (out-degree) in DBLP.

However, despite the above remarks, there still remains a very good sample for our ranking purposes as there is no lack of data with respect to the highly competitive conferences under consideration. Tables 5 and 6 show in a detailed manner the conferences and journals respectively, which have their citations stored, how many citations exist for each one of them, how many of these citations point into the DBLP digital library, how many papers each journal or conference comprises of and, finally, the respective time period. In any case, it is very difficult to collect all the citations to a specific publication. As shown in Figure 6, the number of citations to papers published during the period 1986-1994 is quite high. This gives a good sample for the specific period of time. Unfortunately, the number of citations decreases for years after 1995. Therefore, we will not present the rank results for papers published after 1995, since the sample may be considered inadequate.

Year	Title	P	S	H	C
1986	Object and File Management in the EXODUS Extensible Database System. Michael Carey, David DeWitt, Joel Richardson, Eugene Shekita	2	3	1	2
1987	The R+-Tree: A Dynamic Index for Multi-Dimensional Objects. Timos Sellis, Nick Roussopoulos, Christos Faloutsos	1	1	1	1
1988	Disk Shadowing. Dina Bitton, Jim Gray	1	1	5	2
1989	ARIES/NT: A Recovery Method Based on Write-Ahead Logging for Nested Transactions. Kurt Rothermel, C. Mohan	6	7	14	6
1990	Deriving Production Rules for Constraint Maintainance. Stefano Ceri, Jennifer Widom	1	1	3	1
1991	A Transactional Model for Long-Running Activities. Umeshwar Dayal, Meichun Hsu, Rivka Ladin	2	2	24	4
1992	Querying in Highly Mobile Distributed Environments. Tomasz Imielinski, B.R. Badrinath	3	4	43	12
1993	Universality of Serial Histograms. Yannis Ioannidis	6	7	8	5
1994	Fast Algorithms for Mining Association Rules in Large Databases. Rakesh Agrawal, Ramakrishnan Srikant	1	1	4	1
		23	27	103	35

Table 7. Rank positions of publications awarded with the VLDB 10 Year Award.

Our results are separated in two parts. In the first part we apply the algorithms mentioned earlier to rank/evaluate publications. The second part is dedicated to rank/evaluate authors.

4.1 Ranking Publications

It is a very tough task to evaluate a ranking algorithm for scientific publications, since it is rather subjective which ranking algorithm behavior is better. As a criterion to verify that our ranking results are appropriate, we used two well known awards for publications: the 'VLDB 10 Year Award' and the 'SIGMOD Test of Time Award'. We accept that if any algorithm gives high rank positions to the awarded publications, then this algorithm can be safely used for the task of evaluating publications.

We compare four rank methods: PageRank, SCEAS Rank, HITS Rank (Authorities) and finally the number of Citations. Particular SCEAS Rank variations are not presented here, since they both produce similar results for the data of VLDB and SIGMOD conferences. The tested scenario is the following:

1. Perform all the rank algorithms on the DBLP citation graph.
2. Get only the papers which are inproceedings of VLDB and SIGMOD conferences.
3. Organize and sort the rank tables grouped by conference and year.
4. Check the position of the awarded papers in the above rank tables.

In Tables 7 and 8 we show the awarded publications and their rank position for all of the ranking methods. In these tables, column P stands for PageRank, column S for SCEAS Rank, column H for HITS Rank (Authorities) and column C for the plain Citation counter. For example the paper entitled 'Fast Algorithms for Mining Association Rules in Large DBs, 1994' (Table 7) is ranked first by PageRank, first by SCEAS, fourth by HITS (as an authority) and first by plain Citation counter. Notice again that this way we do not evaluate the awarding committees but the rank methods.

In these detailed tables, we can see that the awarded papers are generally highly ranked. Apparently some deviations and exceptions do exist. These exceptions may exist for two reasons:

- Our citations sample may be not enough: e.g. an awarded publication may get a lot of citations from scientific domains that are not included in the DBLP digital library.

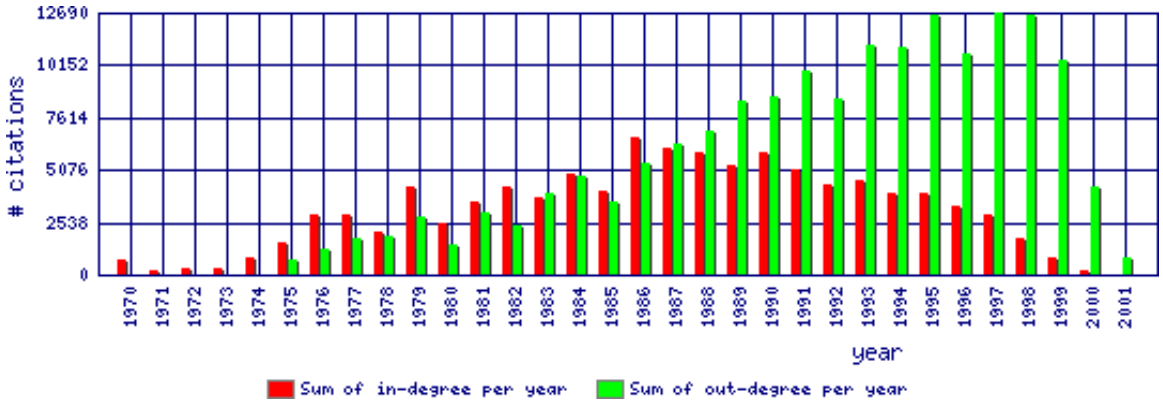


Figure 6. Citations distribution over time (all conferences and journals).

Year	Title	P	S	A	c
1988	A Case for Redundant Arrays of Inexpensive Disks (RAID). David Patterson, Garth Gibson, Randy Katz	1	1	8	2
1989	F-Logic: A Higher-Order language for Reasoning about Objects, Inheritance, and Scheme. Michael Kifer, Georg Lausen	6	4	5	5
1990	Encapsulation of Parallelism in the Volcano Query Processing System. Goetz Graefe	10	9	5	11
1990	Set-Oriented Production Rules in Relational Database Systems. Jennifer Widom, Sheldon Finkelstein	3	3	4	3
1992	Extensible/Rule Based Query Rewrite Optimization in Starburst. Hamid Pirahesh, Joseph Hellerstein, Waqar Hasan	1	1	2	2
1992	Querying Object-Oriented Databases. Michael Kifer, Won Kim, Yehoshua Sagiv	2	3	1	2
1993	Mining Association Rules between Sets of Items in Large Databases. Rakesh Agrawal, Tomasz Imielinski, Arun Swami	1	1	6	1
1994	From Structured Documents to Novel Query Facilities. Vassilis Christophides, Serge Abiteboul, Sophie Cluet, Michel Scholl	2	2	7	2
1994	Shoring Up Persistent Applications. Michael Carey, David DeWitt, Michael Franklin, Nancy Hall, Mark McAuliffe, Jeffrey Naughton, Daniel Schuh, Marvin Solomon, C.K. Tan, Odysseas Tsatalos, Seth White, Michael Zwilling	1	1	1	1
		27	25	39	29

Table 8. Rank positions of publications awarded with the SIGMOD Test of Time Award.

- The awards may be by definition subjective: e.g. an awards committee decision may be based on objective factors (such as citations to papers) but also may combine other measures and indicators of impact.

To get an overall view of the performance of the rank methods, we sum the positions of the awarded publications in the last row. The smaller this sum, the better performance of the ranking method. In both tables, we remark that the HITS Ranking (Authorities) is by far the worst method. This verifies our remarks explained in Section 2. Among the other three methods, the plain citation count is the weakest method, but it still remains a quite good evaluation method. Finally, we see that PageRank and SCEAS Rank are very close to each other and alternate at the winning position in the two Tables 7 and 8. In the sequel, we will ignore the HITS Rank (Authorities) in our tests, since it is not suitable for our evaluation purposes.

Since PageRank, SCEAS Rank and Citation count ended up with about similar results, we will try to produce a single rank table by averaging their results along the reasoning of [11]. In simple words, we will compute all the three rankings and assign to each paper a number of points (5 to 1) depending on its position in the specific rank table. For example, in each table the first paper gets 5 points, the second one gets 4 points, and so on. Thus, if a paper is ranked first in all three rankings, then it will get 15 points. Therefore, we repeat steps 3 and 4 of the previous scenario to produce the new rank tables. This computation is shown in Tables 9 and 10 for the awarded publications. It can be easily seen that the majority of the awarded publications are ranked in the top 3 positions of these new rank tables. Also, after exhaustive experiments we concluded that the sum of the positions (column Pos in both tables) is smaller by using this averaged approach in comparison to using any stand alone rank method. In particular, we remark that in the SIGMOD case (Table 10) a smaller sum of positions is achieved (i.e. 24) in comparison to the VLDB case (Table 9) where the sum is greater (i.e. 27) largely due to the 1989/1999 outlier.

Year	Title	Pos	Points
1986	Object and File Management in the EXODUS Extensible Database System. Michael Carey, David DeWitt, Joel Richardson, Eugene Shekita	2	11
1987	The R+-Tree: A Dynamic Index for Multi-Dimensional Objects. Timos Sellis, Nick Roussopoulos, Christos Faloutsos	1	15
1988	Disk Shadowing. Dina Bitton, Jim Gray	1	14
1989	ARIES/NT: A Recovery Method Based on Write-Ahead Logging for Nested Transactions. Kurt Rothermel, C. Mohan	9	0
1990	Deriving Production Rules for Constraint Maintainance. Stefano Ceri, Jennifer Widom	1	15
1991	A Transactional Model for Long-Running Activities. Umeshwar Dayal, Meichun Hsu, Rivka Ladin	2	10
1992	Querying in Highly Mobile Distributed Environments. Tomasz Imielinski, B.R. Badrinath	4	5
1993	Universality of Serial Histograms. Yannis Ioannidis	6	1
1994	Fast Algorithms for Mining Association Rules in Large Databases. Rakesh Agrawal, Ramakrishnan Srikant	1	15

Table 9. Sum of rank positions of publications awarded with the VLDB 10 Year Award.

4.2 Ranking Authors

We may rely on our method of computing scores for publications and compute scores for authors as well. An approach could be to compute the average score of all their publications. This is not a trivial task. For example, author A has 200 publications with only 40 ones being 'first class'. Assume that these high quality publications have a score of 10 points each, whereas the remaining ones have a score of 1 point. Author B has in total 20 publications, with 10 publications of them being 'first class'. It is reasonable to consider that author A should be ranked higher than author B for his scientific contribution, because A has 4 times the number of first class publications than author B. However, if we just compute the average of all publication scores, then authors A and B would have 2.8 and 5.5 points respectively. Therefore, it is not fair to take into account all the publications a person has authored. In addition, it is not fair to take into account different number of publications

Author Name	Rank Position by average of top-x publications of each author															
	1	2	3	4	5	6	7	8	9	10	12	15	20	25	30	40
C. Mohan	112	122	108	93	83	81	71	69	65	62	54	49	42	34	32	26
David DeWitt	33	26	21	16	15	14	13	13	12	12	12	7	7	5	5	3
David Maier	43	29	29	27	21	19	17	17	16	16	15	13	12	11	12	8
Donald Chamberlin	9	8	4	4	4	4	4	4	4	4	4	4	5	7	8	
Hector Garcia-Molina	88	53	46	41	38	34	33	30	29	27	24	22	22	19	18	12
Jim Gray	3	4	3	2	2	2	2	2	2	2	2	2	2	2	1	1
Michael Stonebraker	24	11	10	9	9	8	8	7	5	5	5	5	4	4	3	2
Patricia Selinger	7	21	23	24	24	23	25	26	25	26	27	32	33	32	34	
Philip Bernstein	63	36	28	21	18	15	14	14	11	11	11	8	8	8	7	5
Rakesh Agrawal	53	37	35	31	27	24	22	22	20	20	19	17	14	12	11	7
Ronald Fagin	49	34	30	26	22	20	19	18	18	18	17	15	17	14	15	13
Rudolf Bayer	25	24	26	28	29	31	31	33	34	38	38	35	37	37	41	34
Serge Abiteboul	165	91	66	54	48	44	43	41	40	36	31	26	25	22	21	15
Lowest ranking point	165	122	108	93	83	81	71	69	65	62	54	49	42	37	41	34
Sum of ranking points	674	496	429	376	340	319	302	296	281	277	259	235	228	207	208	

Table 11. SCEAS Rank average scores for authors.

Year	Title	Pos	Points
1988	A Case for Redundant Arrays of Inexpensive Disks (RAID). David Patterson, Garth Gibson, Randy Katz	1	14
1989	F-Logic: A Higher-Order language for Reasoning about Objects, Inheritance, and Scheme. Michael Kifer, Georg Lausen	5	3
1990	Encapsulation of Parallelism in the Volcano Query Processing System. Goetz Graefe	7	0
1990	Set-Oriented Production Rules in Relational Database Systems. Jennifer Widom, Sheldon Finkelstein	3	9
1992	Extensible/Rule Based Query Rewrite Optimization in Starburst. Hamid Pirahesh, Joseph Hellerstein, Waqar Hasan	1	14
1992	Querying Object-Oriented Databases. Michael Kifer, Won Kim, Yehoshua Sagiv	3	11
1993	Mining Association Rules between Sets of Items in Large Databases. Rakesh Agrawal, Tomasz Imielinski, Arun Swami	1	15
1994	From Structured Documents to Novel Query Facilities. Vassilis Christophides, Serge Abiteboul, Sophie Cluet, Michel Scholl	2	12
1994	Shoring Up Persistent Applications. Michael Carey, David DeWitt, Michael Franklin, Nancy Hall, Mark McAuliffe, Jeffrey Naughton, Daniel Schuh, Marvin Solomon, Tan, Odysseas Tsatalos, Seth White, Michael Zwilling	1	15

Table 10. Sum of rank positions of publications awarded with the SIGMOD Test of Time Award.

for each author (e.g. 40 publications for A and 10 for B). In our approach, we take into account the same number of publications for all authors so that the score results could be comparable.

Therefore, our problem now is to choose the number of publications of each author that should be considered in the ranking. To determine this number we performed the following experiment. We computed the average score for each author by using his top 1-10, ..., 40 publications. Thus, we produced 16 rankings for every rank method. As a testbed we used the authors that were awarded the '*SIGMOD Edgar F. Codd Innovations Award*'. The higher these authors were ranked, the better the evaluation was considered. In Table 11 we present the results that were produced by SCEAS Rank. For brevity, we present only the awarded authors and their position for each selected number of 'top' publications. For example, Hector-Garcia Molina is ranked 88th in the ranking, if the score is produced by the average of 1-best publication of each author. He is ranked 53rd in the ranking, if the score is produced by the average of 2-best publications, and so on.

The last two rows of Table 11 show the rank position of the awarded author that ranked last ('lowest ranking point') and the 'sum of ranking positions' of all the awarded authors. These two numbers are our metrics for comparing the rankings. The lower these numbers are, the best ranking is performed. The lowest ranking point is low indeed when computing the average for the 1-4 best publications of each author. That is due to the fact that the co-authors of a 'high class' publication take advantage and climb up the ranking results. Therefore, by increasing the number of the selected 'best' publications, the awarded authors move towards the top of the rank table. This trend holds until the number of the selected publications becomes 25. The same remark holds when we consider the notion of 'sum of ranking positions'. Also note that in column 40 we miss 2 of the awarded authors, since they have less than 40 publications in the DBLP digital library. For brevity, we have not included the respective tables for the other rank methods, since the results are similar and the smallest ranking point is the same. Thus, after this experiment we decided to rank the authors by averaging their 25 best publications. A final remark with respect to this table is the following. Quite interesting and easily explained is the fact that there are several authors whose ranking position gets higher with increasing number of 'best' publications (with S. Abiteboul advancing the most), whereas the opposite holds for a few other authors (e.g. P. Selinger due to her work on System R and R. Bayer due to his work on B-trees and related structures). Jim Gray steadily holds top positions.

In Table 12 we compare the various rank methods. In this table we present the rank positions of the awarded authors for each Rank method by taking into account the average score of the 25 best publications of each author. In this table column P stands for PageRank, H for HITS Authorities, C for Citation count and S1 and S2 for SCEAS1 and SCEAS2 respactively. Column S1 of this table, is obviously equal to column '25' of Table 11. Similarly to the previous table, we compute the 'Lowest Ranking Point' and 'Sum of ranking points'. In this table we can see that HITS authorities is by far the worst method again, since the 'sum of ranking points' and the 'lowest ranking point' are about 2 and 3 times greater than the respective numbers computed for the other methods. Plain citation count gives an acceptable 'Sum of ranking points', but it fails in the 'lowest ranking point' metric. Finally, SCEAS1 and SCEAS2 are the best methods with respect to the 'lowest ranking point' metric and competitive to PageRank based on the 'Sum of ranking points' metric.

5. Conclusion

In this report we proposed and experimentally examined SCEAS Rank, a new alternative method for scientific publications evaluation, besides the known algorithms of PageRank and HITS. We also

	P	S1	H	S2	C
C. Mohan	41	34	62	33	29
David DeWitt	8	5	4	5	1
David Maier	13	11	8	11	9
Donald Chamberlin	5	7	11	7	14
Hector Garcia-Molina	20	19	115	19	21
Jim Gray	2	2	9	2	3
Michael Stonebraker	4	4	2	4	2
Patricia Selinger	28	32	24	34	38
Philip Bernstein	6	8	6	8	6
Rakesh Agrawal	16	12	82	12	15
Ronald Fagin	10	14	17	14	17
Rudolf Bayer	24	37	117	40	73
Serge Abiteboul	23	22	16	21	13
Lowest Ranking point	41	37	117	40	73
Sum of ranking points	200	207	473	210	241

Table 12. Rank position of awarded authors by average of 25 best publications.

presented SCEAS Rank tuned variations. However, detailed algorithm descriptions and performance tuning appears in [13]. We also evaluated the above method by using the DBLP digital library as a training set and the awarded publications of '*VLDB 10 Year Award*' and '*SIGMOD Test of Time Award*' as an evaluation set for the publications rank method. Additionally, we presented author ranking based on the publication rank results and we used the '*SIGMOD Edgar F. Codd Innovations Award*' as an evaluation set. In both cases the performance of SCEAS Rank was in general better than the other methods. This might be helpful to short-list candidate authors for awarding during the next few years.

References

[1] S. Brin and L. Page. The Anatomy of a Large-scale Hypertextual Web Search Engine. In *Proceedings 7th WWW Conference*, pages 107–117, 1998.

[2] S. Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*, chapter 7.1, pages 205–206. Morgan Kaufmann Publishers, 2003.

[3] E. Garfield. Science citation index. http://www.isinet.com/isi/.

[4] E. Garfield. Citation Analysis as a Tool in Journal Evaluation. *Essays of an Information Scientist*, 1:527–544, 1972.

[5] E. Garfield. The Impact Factor, 1994. http://www.isinet.com/isi/hot/essays/journalcitationreports/7.html.

[6] S. D. Kamvar and T. H. Haveliwala. The condition number of the pagerank problem. Technical report, Stanford University, 2003.

[7] J. Kleinberg. Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM*, 46(5):604–632, 1999.

[8] J. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. The Web as a Graph: Measurements, Models, and Methods. In *Proceedings 5th COCOON Conference*, pages 1–17, 1999.

[9] J. Kleinjnen and W. V. Groenendaal. Measuring the Quality of Publications: New Methodology and Case Study. *Information Processing and Management*, 36(4):551–570, 2000.

[10] N. Mylonopoulos and V. Theoharakis. Global Perceptions of IS Journals. *Communications of the ACM*, 44(9):29–33, 2001.

[11] R. Rainer and M. Miller. Examining Differences across Journal Rankings. *Communications of the ACM*, 48(2):91–94, 2005.

[12] A. Sidiropoulos and Y. Manolopoulos. A New Perspective to Automatically Rank Scientific Conferences Using Digital Libraries. *Information Processing and Management*, 41(2):289–312, 2005.

[13] A. Sidiropoulos and Y. Manolopoulos. Ranking Algorithms for Publications and Authors. Technical report, Aristotle University, 2005. under submission.

An Apples-to-Apples Comparison of Two Database Journals

Philip A. Bernstein, Microsoft Research
Elisa Bertino, Purdue University
Andreas Heuer, University of Rostock
Christian S. Jensen, Aalborg University
Holger Meyer, University of Rostock
M. Tamer Özsu, University of Waterloo
Richard T. Snodgrass, University of Arizona
Kyu-Young Whang, KAIST

Abstract

This paper defines a collection of metrics on manuscript reviewing and presents historical data for *ACM Transactions on Database Systems* and *The VLDB Journal*.

1. Introduction

The editors of *TODS* and *The VLDB Journal* have collaborated to generate historical data based on common definitions of relevant metrics. The data reported here was first presented, in preliminary form, at a VLDB panel [1]. This data has been updated in the intervening time to fix inconsistencies and improve clarity.

We found this to be a useful process, as the underlying data was cleaned and as we were able to observe some trends in metrics that had never before been computed. Additionally, we feel that such historical data is important for the community, authors, editors, and readers alike. The most recent data on manuscript processing times and rates is of interest to potential authors, some of the metrics are of interest to readers in judging the timeliness of the material published by the journal, and the historical trends are of interest to the database community at large, as it helps us to understand how scientific publishing is evolving.

This paper begins with a definition of nine metrics. Historical data for *TODS* and for *VLDB J.* for these metrics is given in the following two sections. We conclude with some possible next steps.

2. Manuscript Flow Model and Journal Metrics

The process flow model is oriented around the events that submitting authors' and subsequent readers' experience.

The following is a high-level summary of the manuscript review process used by most database journals. A *round* is one of the following: either (a) a manuscript (original submission or revision) is submitted and an editorial decision is made (i.e., accept, reject, or minor/major revision) after gathering one or more external reviews, or (b) a manuscript is submitted and an editorial decision is made without going to reviewers. A manuscript that is accepted and subsequently published and appears in an issue of the journal is termed an *article*. Manuscripts that are withdrawn during the first round are not counted as submissions. Manuscripts that are withdrawn at some point after the first round are counted as rejections.

There are 9 metrics. Metrics 1–4 and 8 include average and maximum figures, and minimum figures when meaningful numbers are available.

1. *First-round turnaround time*: The time for the first round, measured from the manuscript's submission date to the journal to the date that an editorial decision is sent to the author(s). The X axis is the year of the submission. We report the average for that year and the maximum, both in months. The minimum is not reported because these make little sense due to desk rejects.
2. *Overall turnaround time*: Same as first-round turnaround, but measured for *all rounds* that were initiated in a given year (i.e., for both original submissions and revisions). The X axis is the year the round was initiated. The overall turnaround time is generally shorter than the first-round, because revisions are sent to the reviewers of the original manuscript. Again, average and maximum, in months, for that year are reported.
3. *Acceptance Time*: The difference between the date of the accept decision and the date of initial submission, in months. The X axis is the year of the initial submission. We only report on years in which all submissions have been finalized (accepted or rejected), and report the average per year, the minimum, and the maximum.
4. *End-to-end time*: The difference between the date of the issue in which the article eventually appeared and the date of the initial submission. The X axis is the year of publication (note that this differs from acceptance time, which is based on the year of the initial submission). The date of an issue is generally a month; if it is a range such as January/March, then the last date is used as the issue date.
5. *Number of submissions*: The absolute number of submitted manuscripts in each year.
6. *Acceptance rate*: The percentage of those manuscripts submitted that year that were ultimately accepted. The X axis is the year the manuscript was initially submitted. Only years for which all submitted

manuscripts have been accepted or rejected (that is, are not still in review or revision) are included.

We also provide some publication metrics. In metrics 7–9, the X axis is the year the volume covered (for most of the time, a volume represented the issues published that year).

7. *Number of articles per volume*.
8. *Article length per volume*: The number of formatted pages of an article, including bibliography and printed appendices, but not electronic-only appendices. We report the average article length and the length of the shortest and longest article that appeared that year.
9. *Total page length per volume*: The sum of the lengths of the articles of that volume.

We did not include metrics that relate to internal journal processes that are not visible to authors and readers, such as editor responsiveness, reviewer responsiveness, and number of reviews originally requested.

3. ACM *Transactions on Database Systems*

Metrics 1 and 2 are provided in Figure 1, metric 3 is in Figure 2, and metric 4 is in Figure 3. Some of these metrics are shown as tables due to lack of detailed data. For manuscripts for which only the submission or acceptance *month* was known (all papers prior to 2001 and eight papers after 2001), the first day of the month was assumed. The data points in Figure 3 do not include about 23 papers during 1976–1998, about one a year, for which detailed data is not known. Almost all accepted papers go through two rounds of reviewing and a revision, the latter averaging around four months.

Metrics 5 and 6 are shown in Figure 4. The acceptance rate for 2004 is not given, as there are seven submissions still in review (all as revisions). Metrics 7–9 are in Figures 5–7. There was one volume per year (four issues, March, June, September, and December).

	First Round		Overall	
Year	Avg	Max	Avg	Max
2002	3.4	7.1	3.4	7.1
2003	2.9	6.5	2.9	6.5
2004	3.0	5.0	2.9	5.0

Figure 1 *TODS* Turnaround Time in Months

Year Initially Submitted	Min	Avg	Max
2002	5.5	12.8	31.7
2003	4.0	10.1	19.5

Figure 2 *TODS* Acceptance Time in Months

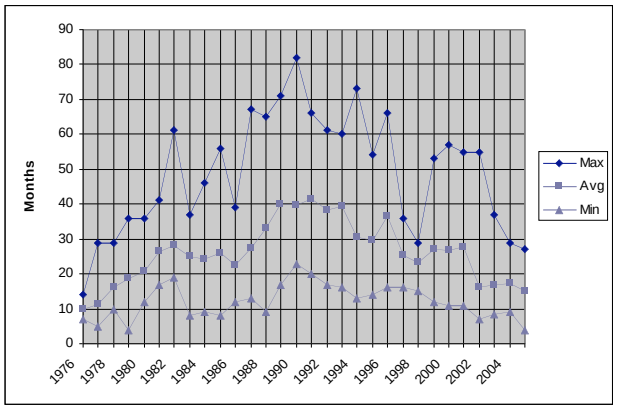


Figure 3 *TODS* End-to-End Time

Year	Number Submitted	Acceptance Rate
2002	60	38%
2003	72	36%
2004	79	---

Figure 4 *TODS* Submission and Acceptance Rate

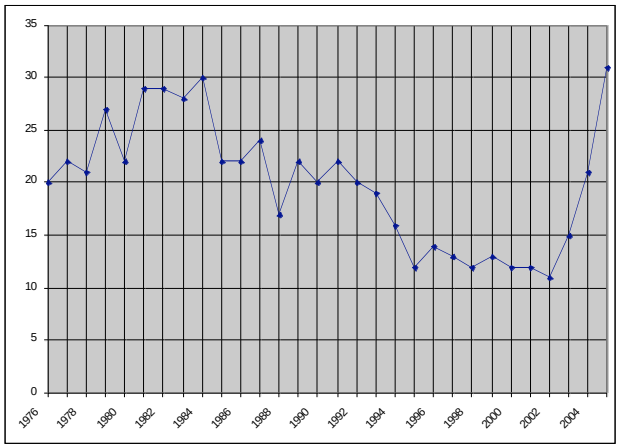


Figure 5 *TODS* Number of Articles per Volume

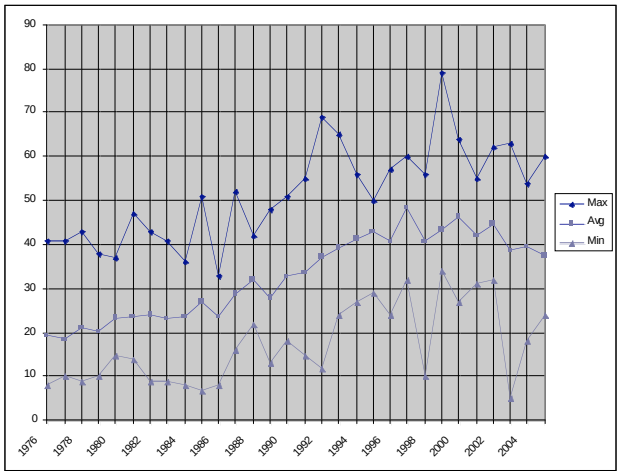


Figure 6 *TODS* Article Length per Volume

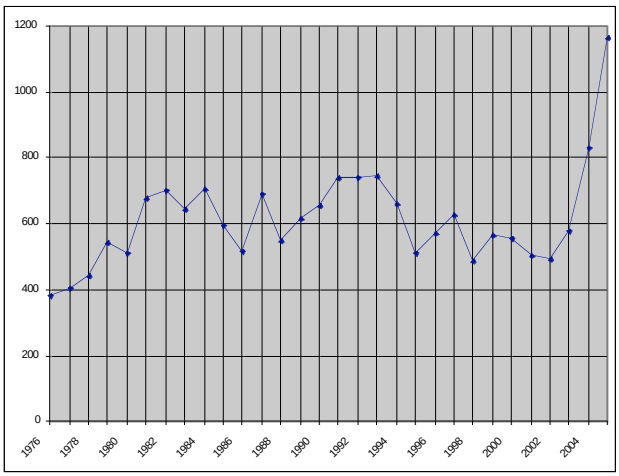


Figure 7 *TODS* Total Pages per Volume

4. *The VLDB Journal*

Metrics 1 and 2 are in Figure 8. Metrics 3–9 are in Figures 9–15 respectively.

- In Figures 8 and 9, we have reported the median instead of the average because that is the way such data has traditionally been reported to the VLDB Endowment's Board of Directors.
- Figure 10 does not include minimums because that data is not available.
- Overall turnaround times are measured for the first and the second rounds, but not for the third round, due to lack of detailed data. If the date of the revised submission is not available, we have used the first decision date (i.e., an upper bound) instead for calculating the second round turnaround time.
- One paper submitted in 2003 is pending in the second round; four submitted in 2004 are pending in the second round and two in the third round.
- In 1996, *VLDB J.* moved from Boxwood Press (roughly the *TODS* page format) to Springer-Verlag (in a larger format). We estimate the latter's page size as 1.86 times the *TODS* page size. Figures 14 and 15 show curves normalized to the *TODS* format based on that factor.
- In Figures 13–15, years do not map exactly to volumes, e.g., for 1999 and 2000, when final issues of a volume were published late.

5. Next Steps

It would be useful to refine these metrics, based on input from the community. For example, are averages, medians, or both preferred? Should rounds that don't go to reviewers, and thus are much quicker, be differentiated from rounds that utilize reviewers? What other metrics would be useful? One possibility is the *reference age*, the average interval from the citation of the most recent published paper to the print publication date [2].

It would also be useful to enlist additional database journals in this exercise.

References

1. Philip A. Bernstein, David DeWitt, Andreas Heuer, Zachary Ives, Christian S. Jensen, Holger Meyer, M. Tamer Özsu, Richard T. Snodgrass, Kyu-Young Whang, and Jennifer Widom, Database Publication Practices—Panel. VLDB 2005:1241–245. http://www.vldb2005.org/program/manuscript/web/p1241-bernstein.pdf
2. Richard T. Snodgrass, Journal Relevance, *SIGMOD Record* 31(3):11–15, September, 2003.

	First Round		Overall	
Year Submitted	Med	Max	Med	Max
2002	4.1	13.0	4.0	14.8
2003	5.1	13.9	3.9	13.9
2004	4.6	12.8	4.0	12.8

Figure 8 *VLDB J.* Turnaround Time in Months

Year Initially Submitted	Min	Med	Max
2002	5.2	9.1	30.0
2003	5.9	8.1	20.6

Figure 9 *VLDB J.* Acceptance Time in Months

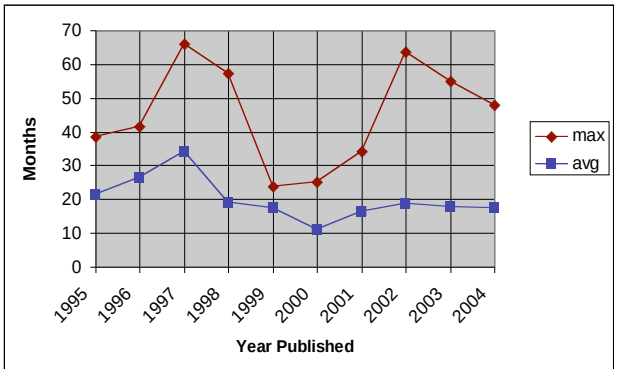


Figure 10 *VLDB J.* End-to-End Time

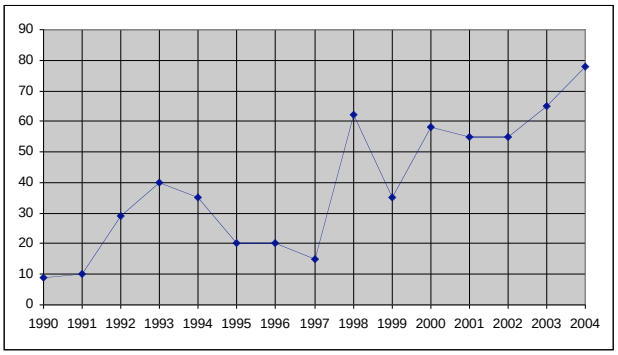


Figure 11 *VLDB J.* Number of Submissions

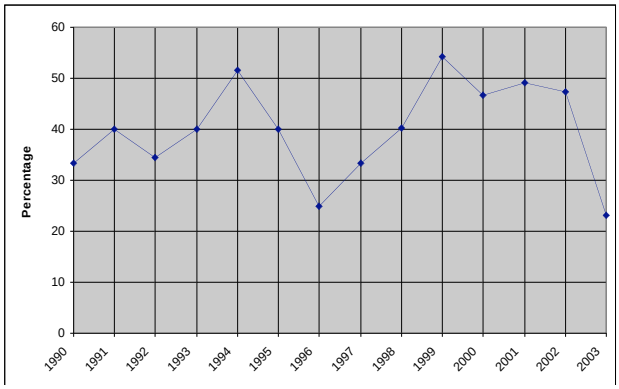
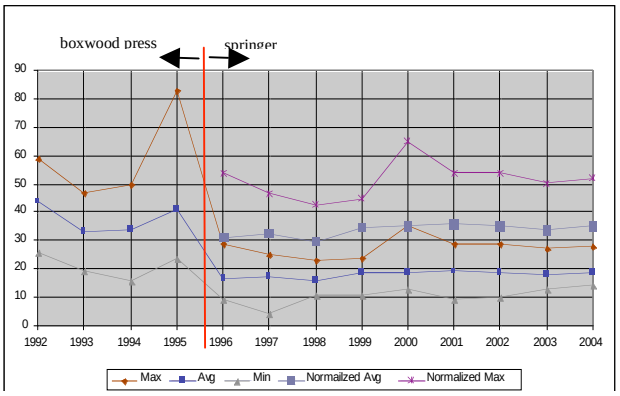


Figure 12 *VLDB J.* Acceptance Rate



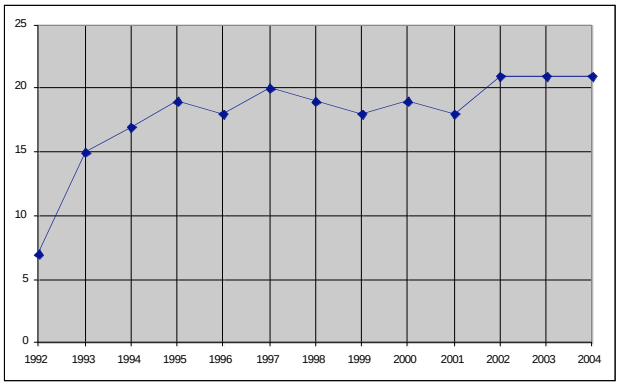


Figure 13 *VLDB J.* Number of Articles per Year

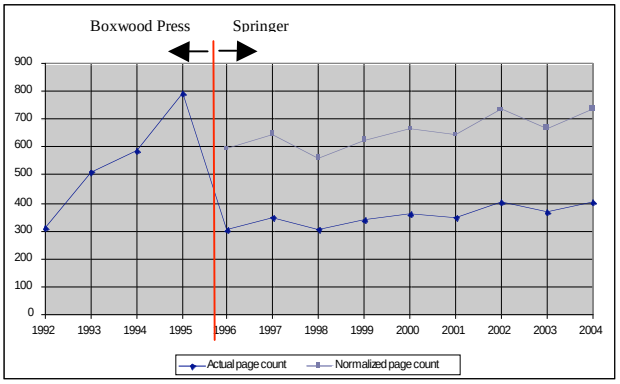


Figure 15 *VLDB J.* Total Pages per Year

Data Management Research at Technische Universität Darmstadt

A. Buchmann and M. Cilia
Databases and Distributed Systems Group, Department of Computer Science
Technische Universität Darmstadt - Darmstadt, Germany
<lastname>@dvs1.informatik.tu-darmstadt.de

Abstract

The Databases and Distributed Systems Group at Technische Universität Darmstadt is devoted to research in the areas of data management middleware and reactive, event-based systems. Special emphasis is placed on handling the flow of data and events in a variety of environments: publish/subscribe mechanisms, information dissemination and integration, ubiquitous computing, peer-to-peer infrastructures, and a variety of sensor-based systems ranging from passive RFID infrastructures to active wireless sensor networks. A special concern is placed on non-functional aspects of the middleware, such as performance, scalability and security, where members of our group are involved in the definition of the SPEC family of benchmarks for J2EE (SPECjAppServer200x) and JMS.

1 Introduction

The work in the group is based on several observations derived from the convergence of technologies:

- the deployment of smart devices requires the continuous monitoring of events and context data for their correct interpretation;
- the miniaturization of sensors and their ubiquitous deployment will result in massive amounts of sensor data that must be processed, often under real-time constraints;
- the heterogeneity of resources of the participating nodes and their unstable connectivity leads to mixed-mode systems with special needs with respect to consistency, availability, security, etc.;
- huge distributed systems must be capable of detecting and correcting failures and return autonomously to stable operation;
- new business strategies, such as event-driven supply chain management and zero-latency enterprises, depend on the timely dissemination of information and business events.

The basic premise underlying our work is that information flows and is no longer confined to repositories. Therefore, traditional pull-based access mechanisms to stagnant data are no longer sufficient. In addition, a reliable infrastructure for management of streams of data and events is needed. The importance of this infrastructure will increase as we move to a world populated by huge amounts of interconnected devices with different capabilities that will react and automate processes on our behalf.

This research survey is organized into 3 sections: Data Dissemination, Peer-to-Peer meets Pub/Sub, and Performance modeling.

2 Data Dissemination

Data dissemination is one of the core problems in monitoring applications ranging from RFID-based logistics and warehouse control to ambient intelligence. We have developed solutions based on the Publish/Subscribe paradigm and have extended the basic content-based routing to accommodate heterogeneity, and developed composition mechanisms for subscriptions (filters) and notifications (data and events). We have addressed quality of service and management issues through the development of middleware-mediated transactions and scopes. We provided for reactive capability that is modular and composable at the subscriber end, and are currently extending the publish/subscribe notification system to accommodate mobility [14], deal with context information [29], and provide some basic security [13].

2.1 Routing strategies

We developed a notification service framework called REBECA. It basically offers a distributed event notification service to which applications and other system services are connected as clients. These clients act as producers and consumers of notifications. The notification service itself is an overlay network in the underlying system, consisting of a subset of nodes connected in a network of event brokers. The brokers receive notifications, filter and forward them in order to deliver published notifications to all attached consumers having a matching subscription.

We carried out several experiments on top of REBECA. They show that in large-scale systems, more advanced content-based routing algorithms must be applied [23]. Those algorithms exploit commonalities among subscriptions in order to reduce routing table sizes and message overhead. We have investigated three of them, identity-based routing, covering-based routing [6], and merging-based routing [22]. Identity-based routing avoids forwarding of subscriptions that match identical sets of notifications. Covering-based routing avoids forwarding of those subscriptions that only accept a subset of notifications matched by a formerly forwarded subscription. Note that this implies that it might be necessary to forward some of the covered subscriptions along with the unsubscription if a subscription is cancelled. Merging-based routing goes even further. In this case, each broker can merge existing routing entries to a broader subscription, i.e., the broker creates new covers.

Advertisements can be used as an additional mechanism to further optimize content-based routing. They are filters that are issued (and cancelled) by producers to indicate (and revoke) their intention to publish certain kinds of notifications. If advertisements are used, it is sufficient to forward subscriptions only into those subnets of the broker network in which a producer has issued an overlapping advertisement, i.e., where matching notifications can be produced. Advertisements can be combined with all routing algorithms discussed above.

2.2 Filters and composition

Events can be either primitive or composite. In most practical situations primitive events, e.g. events detected by basic sensors or produced by applications, must be combined. Usually, this composition or aggregation relies on an event algebra that may include operators for sequence, disjunction, conjunction, etc.

However, these event algebras and consumption policies depend on a total order of events and are based on point-based timestamps of a single central clock. These assumptions are invalidated by the inherent characteristics of distributed environments. Therefore, the event occurrence time must be considered to be indeterminate to some extent. As a consequence, time indeterminacy must be reflected in the time model and explicitly recognized and reported when composing events in distributed and heterogenous systems [19].

We have built an event aggregation service that is based on the principles of components and containers [9]. Containers control the event aggregation process while components define the event operators logic. As mentioned before, the aggregation service is treated like any other event consumer that can subscribe to events, it aggregates them and finally publishes the aggregated event. The handling of time indeterminacy and network delays are encapsulated in such a container.

Additionally, in many cases the traffic of messages within a notification service can be reduced by applying filters. For this purpose a framework for filter definition is under construction. These filters can simply discard events according to some pattern (one in ten), or by placing them close to their source where a straightforward analysis of relevant changes (for instance, the analysis of regular events signaled by a temperature sensor) is carried out.

2.3 Data integration

In a realistic environment events are produced at heterogeneous/diverse sources. These events encapsulate data, which can only be properly interpreted when sufficient context information about its intended meaning is known. In general, this information is left implicit and as a consequence, it is lost when data/events are exchanged across institutional or system boundaries. Combining or interpreting data from different sources leads inevitably to problems if the meaning of terms is not shared.

In the context of notification services, consumers need to know about the content of the events/messages that are being exchanged in order to express their subscriptions. That means, that consumers must know details about the representation and assumed semantics of message content. Today, notification services do not support this leaving required information about data semantics implicit. Without this information event producers and consumers are expected to fully comply with implicit assumptions made by participants.

The approach we have taken solves this problem by providing a concept-based layer on top of the delivery mechanism [7]. This layer provides a higher level of ab-

straction in order to express subscription patterns and to publish events with the necessary information to support their correct interpretation outside the producers' boundaries. This was achieved by relying on the MIX model [3, 4] for the representation of data (i.e., event content). MIX directly supports data integration by making the concept of semantic context (i.e., the explicit description of implicit assumptions about the meaning of the data) and conversion functions (which allow the automatic conversion of data/events from different sources to a common context) first class citizens of the model itself.

2.4 Multi-hop transaction support

In distributed settings, the application process typically spans multiple transactional information systems. Grouping the information access into a single distributed transaction requires resources to be locked for the duration of the transaction and termination must be coordinated by a 2-phase-commit protocol. While this approach is realized in standardized and commonly applied middleware services [24], the applicability thereof is restricted to tightly coupled systems and thus is not suitable for the integration of autonomous components.

In the event-based architectural style the event producer is decoupled from the event consumer through the mediator. Therefore, any transaction concept in an event-based system must include the mediator. On the other hand, applications will be implemented in some (object-oriented) programming language. The challenge is therefore, to combine notifications with conventional transactional object requests into middleware mediated transactions (MMT) [21]. MMTs extend the atomicity sphere of transactional object requests to include mediators and/or final recipients of notifications.

In order to integrate producers, mediators and subscribers, a more flexible transactional framework was developed [20]. This framework provides the means to couple the visibility of event notifications to the boundaries of transaction spheres and the success or failure of (parts of) a transaction. It also describes the transactional context in which the consumer should execute its actions. It specifies the dependencies between the triggering and the triggered transactions, dynamically spanning a tree of interdependent transactional activities.

2.5 Reactive capabilities

Emerging trends like, event-driven supply chain management, the zero-latency enterprise, or ambient intelligence applications depend on the timely dissemination of data but also on the proper reaction to those events. The Event-Condition-Action rule (ECA-rule) approach fits very well in this context, but does not always require a full-fledged database support. We decomposed the traditional processing of ECA-rules (typically embedded in active databases) into its elementary and autonomous parts [8]. These parts are responsible for event aggregation (see Section 2.2), condition evaluation and action execution. The processing of rules is then realized as a composition of these elementary services on a per rule basis. This composition forms a chain of services that are in charge of processing the rule in question. These elementary services interact among them based on the notification service. As mentioned before, the reactive service is treated like any other event consumer that can subscribe to events. When events of interest (i.e. those that trigger rules) are notified, the corresponding rule processing chain is automatically activated. Elementary services (i.e. action execution) that interact with external systems or services use *plug-ins* for this purpose. Besides that, plug-ins are responsible for maintaining the semantic target context of the system they interact with making possible the meaningful exchange of data. This service is used in the context of online meta-auctions [5], the Internet-enabled car [10] and an RFID-based supply chain scenario as well.

2.6 Scoping

Despite the numerous advantages offered by the loose coupling of event-based interaction, a number of drawbacks arise from the new degrees of freedom. Event systems are characterized by a flat design space in which subscriptions are matched against all published notifications without discriminating producers. This makes event systems difficult to manage. A generic mechanism is needed to control the visibility of events, e.g. for security reasons, and for structuring sets of producers and consumers, extending visibility beyond the transactional aspects (as presented early in Section 2.4). Operational controls and management tasks can then be bound to these structures.

Scopes [12] allow system engineers to exert explicit control on the event-based interaction; it is a functionality orthogonal to the different layers of a notification service. We see scopes as the means by which system administrators and application developers can configure an event-based system. Scopes offer an abstraction to identify structure and to bind organization and control of routing algorithms, heterogeneity support, and transactional behavior to the application structure. They delimit application functionality and contexts, controlling side effects and associating ontologies at well-defined points in the

system. This is of particular importance as platforms of the future must be configurable not only at deployment time but also once an application is in operation. We have introduced scopes in two different environments such as the J2EE platform [11] and Wireless Sensor Networks [27, 26].

3 Peer-to-Peer meets Pub/Sub

The convergence of technologies we alluded to in the introduction and our interest in non-functional properties, such as scalability and robustness, also pushed us to look at the question: what happens if you try to combine the behaviour of a publish/subscribe system with the resilience of a peer-to-peer substrate. This question does not only have academic appeal as can be seen from the gaming scenario we use both as a motivation and for requirements.

3.1 Gaming as a motivation

The gaming industry is about to surpass the movie industry in total revenue. One of the fastest growing branches is the one of massive multi-player online games. MMOGs muster followings of several hundred thousands of players who subscribe on a monthly basis to play over years in a virtual world divided into shards. In the current client/server architectures, technical limitations impose a limit of about 7 000 players per game server. One of the huge intangibles when launching such a game is the success rate. If the success rate is estimated too optimistically, a huge investment in infrastructure is wasted; on the contrary, a pessimistic estimate may lead to sluggish performance and the loss of favour in the gaming community. An interesting solution is to develop MMOGs on a P2P infrastructure. This idea exploits the fact that gamers tend to have state of the art hardware and communications. Migrating MMOGs to a P2P platform implies pushing game events to many servers under controlled conditions and raises many quality of service issues: latency, robustness, scalability, consistency of the game states, security, etc. We have been looking at many of these issues from the perspective of how to control cheating in a gaming environment without central controls [15].

3.2 Building P2P networks with controlled QoS

One of the biggest challenges in the P2P community is to build systems with controlled quality of service. In most cases, P2P systems are laboriously handcrafted and a posteriori their behaviour might be studied. We have approached the building of QoS aware P2P systems from a database point of view [2]. In a first step, nodes with certain quality attributes (e.g. 90% availability) are declaratively selected from a node database. In a second step, a parameterized topology is selected, according to which the nodes will be connected. This tool allows us to configure new P2P networks with different quality attributes and topologies with a few lines of code [1]. While the present system represents progress in the right direction and allows us to easily build P2P systems with nodes of individual QoS characteristics, it is still a long way to predicting global QoS, which is the subject of ongoing work.

3.3 The Rendezvous problem

In many distributed applications, pairs of queries and values are evaluated by participating nodes. Examples include keyword searches for documents, selection queries on tuples, or matching of filters and notifications in publish/subscribe systems. In a distributed system the key question is: where should the evaluation take place and how can data movement be minimized. Work on this generic problem in the P2P context resulted in the bit-zipper approach [28], which deduces from the coding scheme, at which node of a distributed system query and data (or filter and notifications) should optimally meet. The bit-zipper is provenly optimal (in terms of messages sent) for problems in which all pairs of queries and data must be evaluated. Where flooding to N nodes was previously the only fall-back, the bit-zipper needs only $O(\sqrt{N})$. Ongoing research is looking at lower and upper bounds and further generalizations in the processing of queries in P2P systems.

4 Performance Modeling, Analysis and Prediction

Modern applications are typically built on highly distributed, multitiered platforms that are deployed in heterogeneous environments. The complexity of such systems makes it difficult to anticipate the performance of a given deployment. Load testing and benchmarking is quite useful for the identification of bottlenecks, however, it is impractical and costly since a deployment environment like the final application deployment is required. For any kind of extrapolation or decisions earlier in the design cycle, performance models are needed. The Databases and Distributed Systems Group is active in both areas, for tradi-

tional enterprise applications as well as notification services and event-based systems.

4.1 Enterprise applications

As members of the SPEC Java Subcommittee we have been actively involved in developing the SPECjAppServer family of benchmarks for the J2EE platform [25]. This has also allowed us to calibrate and validate our performance models against large-scale deployments of the benchmarks [16, 17].

The performance models that were developed in the group are based on traditional queuing networks [16] and on queuing Petri nets [17]. The QN models are quite accurate for modelling throughput, however, they suffer from certain drawbacks when modelling response time. QN models are suitable for modelling active resources, such as CPUs, but are inadequate to model software contention, as they do not provide any means for modelling synchronization. This problem can be solved by using QPN-based models. QPNs insert queues in the places of a Petri net and provide a good tool for modelling synchronization. However, they suffer from the common problem of state space explosion.

The solution was the development of a simulator based on QPNs. This simulator has been calibrated against analytical models where possible and against a wide range of deployments of the SPECjAppServer2004 benchmark. These deployments include both commercial as well as open source J2EE platforms on individual servers as well as clusters [18].

4.2 Asynchronous interactions

Current performance work is centered on the development of benchmarks and performance models for asynchronous interactions. Members of the group are currently involved in the development of a benchmark for JMS in the context of the SPEC Java Subcommittee. In other cooperation with industry, we evaluated the performance of SAP's AutoID infrastructure. Through these activities we are in a position to experiment with some industry strength RFID deployment scenarios.

Recently we have been making progress on the development of load generation tools for event-based systems. Part of this effort is the visualization of the behaviour of the analyzed platforms as a whole and through introspection of individual components. The latter is achieved through the application of Aspect Oriented Programming techniques that allow us to monitor the internal operation of individual components.

The long-term goal is to develop both analytic models and simulators for event-based asynchronous interactions and calibrate and validate them against a large application scenarios reflected in an industrial strength benchmark.

Acknowledgements

We would like to acknowledge the contributions of the many present and past members of the Databases and Distributed Systems Group who helped shape the present research and whose results we summarized: Jose Antolini, Mario Antolini, Stefan Behnel, Christof Bornhövd, Silvana D'Cristofaro, Ludger Fiege, Cristian Fiorentino, Felix Freiling, Pablo Guerrero, Alejandro Houspanousian, Kai Juse, Dimka Karastoyanova, Roger Kilian-Kehr, Christoph Liebig, Matthias Meixner, Gero Mühl, Diego Palmisano, Kai Sachs, Sebastian Salvucci, Jan Stefan, Wesley Terpstra, Björn Weis, Andreas Zeidler. Thanks go also to previous doctoral students who helped set the stage and the many master students who made their contributions and life so enjoyable.

References

[1] S. Behnel and A. Buchmann. Models and languages for overlay networks. In *Proc. of VLDB Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P 2005)* , Trondheim, Norway, August 2005.

[2] S. Behnel and A. Buchmann. Overlay networks - implementation by specification. In *Proc. of Middleware 2005*, Grenoble, France, November 2005. (To appear).

[3] C. Bornhövd. *Semantic Metadata for the Integration of Heterogeneous Internet Data (in German)*. PhD thesis, Darmstadt University of Technology, 2000.

[4] C. Bornhövd and A.P. Buchmann. A Prototype for Metadata-Based Integration of Internet Sources. In *Proc. of CAiSE*, volume 1626 of *LNCS*, 1999.

[5] C. Bornhövd, M. Cilia, C. Liebig, and A.P Buchmann. An Infrastructure for Meta-Auctions. In *Proc. of WECWIS'00*, June 2000.

[6] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, 2001.

[7] M. Cilia, M. Antollini, C. Bornhvd, and A. Buchmann. Dealing with heterogeneous data in pub/sub systems: The Concept-Based approach. In *Intl Workshop on Distributed Event-Based Systems (DEBS'04)*, May 2004.

[8] M. Cilia, C. Bornhövd, and A. P. Buchmann. Moving Active functionality from Centralized to Open Distributed Heterogeneous Environments. In *Proc. of CoopIS'01*, volume 2172 of *LNCS*, 2001.

[9] M. Cilia, C. Bornhvd, and A. Buchmann. CREAM: an infrastructure for distributed, heterogeneous event-based applications. In *Proc. of CoopIS'03*, volume 2888 of *LNCS*, pages 482–502, Catania, Italy, November 2003. Springer.

[10] M. Cilia, P. Hasselmeyer, and A.P. Buchmann. Profiling and Internet Connectivity in Automotive Environments. In *Proc. of VLDB'02*, August 2002.

[11] Dan Dobre. A framework for engineering J2EE-based publish/subscribe applications with scopes. Master's thesis, Technische Universitt Darmstadt, Dept of Computer Science, Germany, 2004.

[12] L. Fiege, M. Mezini, G. Mühl, and A.P. Buchmann. Engineering event-based systems with scopes. In *Proc. of ECOOP'02*, volume 2374 of *LNCS*, 2002.

[13] L. Fiege, A. Zeidler, A. Buchmann, R. Kilian-Kehr, and G. Mhl. Security aspects in publish/subscribe systems. In *Workshop on Distributed Event-based Systems (DEBS'04)*, May 2004.

[14] Ludger Fiege, Felix C. Grtner, Oliver Kasten, and Andreas Zeidler. Supporting mobility in Content-Based publish/subscribe middleware. In *Proc. of Middleware 2003*, pages 103–122, June 2003.

[15] P. Kabus, W. Terpstra, M. Cilia, and A. Buchmann. Addressing Cheating in Distributed Massively Multiplayer Online Games. In *Proc. of Intl Workshop on NetGames*, October 2005.

[16] S. Kounev and A. Buchmann. Performance Modeling and Evaluation of Large-Scale J2EE Applications. In *Proc. of Intl Conf of the Computer Measurement Group (CMG) on Resource Management and Performance Evaluation of Enterprise Computing Systems*, December 2003.

[17] S. Kounev and A. Buchmann. Performance Modelling of Distributed E-Business Applications using Queuing Petri Nets. In *Proc. of the IEEE Intl Symp on Performance Analysis of Systems and Software (ISPASS'03)*, March 2003.

[18] S. Kounev and A. Buchmann. SimQPN - a tool and methodology for analyzing queueing Petri net models by means of simulation. *Performance Evaluation Journal*, 2006. (To appear).

[19] C. Liebig, M. Cilia, and A.P. Buchmann. Event Composition in Time-dependent Distributed Systems. In *Proc. of CoopIS'99*, 1999.

[20] C. Liebig, M. Malva, and A.P. Buchmann. Integrating Notifications and Transactions: Concepts and X^2TS Prototype. In *Proc. of EDO*, LNCS 1999, 2000.

[21] Christoph Liebig and Stefan Tai. Middleware mediated transactions. In *Proc. of DOA'00*, 2001.

[22] G. Mühl, L. Fiege, and A.P. Buchmann. Filter Similarities in Content-Based Publish/Subscribe Systems. In *Proc. of ARCS*, LNCS 2299, 2002.

[23] G. Mühl, L. Fiege, F.C. Gärtner, and A.P. Buchmann. Evaluating advanced routing algorithms for content-based publish/subscribe systems. In *Proc. IEEE/ACM MASCOTS'02*, 2002.

[24] Object Management Group (OMG). Transaction service v1.1. Technical Report OMG Document formal/2000-06-28, OMG, May 2000.

[25] SPEC. The SPECjAppServer2004 Project, 2004. `www.spec.org/jAppServer2004`.

[26] J. Steffan, L. Fiege, M. Cilia, and A. Buchmann. Scoping in wireless sensor networks. In *Intl Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC'04)*, October 2004.

[27] J. Steffan, L. Fiege, M. Cilia, and A. Buchmann. Towards Multi-Purpose wireless sensor networks. In *Proc. of Conf. on Sensor Networks (SENET'05)*, August 2005.

[28] W. Terpstra, S. Behnel, L. Fiege, J. Kangasharju, and A. Buchmann. Bit Zipper Rendezvous – Optimal Data Placement for General P2P Queries. In *EDBT 04 Workshop on Peer-to-Peer Computing & DataBases*, March 2004.

[29] A. Zeidler. *A Distributed Publish/Subscribe Notification Service for Pervasive Environments*. Ph.D. Thesis, Department of Computer Science, Darmstadt University of Technology, Germany, 2004.

Report on the DB/IR Panel at SIGMOD 2005

Sihem Amer-Yahia
Moderator
AT&T Labs Research, USA

Pat Case
Library of Congress, USA
Thomas Rölleke
Queens Mary University, UK
Jayavel Shanmugasundaram
Cornell University, USA
Gerhard Weikum
Max Plank Institute, Germany

1. MOTIVATION

This paper summarizes the salient aspects of the SIGMOD 2005 panel on "Databases and Information Retrieval: Rethinking the Great Divide". The goal of the panel was to discuss whether we should rethink data management systems architectures to truly merge Database (DB) and Information Retrieval (IR) technologies. The panel had very high attendance and generated lively discussions.[1]

Until now, the DB and IR communities, while each very successful, have evolved largely independently of each other. The DB community has mostly focused on highly structured data, and has developed sophisticated techniques for efficiently processing complex and precise queries over this data. In contrast, the IR community has focused on searching unstructured data, and has developed various techniques for ranking query results and evaluating their effectiveness. Consequently, there has been no single unified system model for managing both structured and unstructured data, and processing both precise and ranked queries. Most prior integration attempts have "glued" together DB and IR engines without making fundamental changes to either engine.

However, emerging applications such as content management and XML data management, which have an abundant mix of structured and unstructured data, require us to rethink data management assumptions such as the strict dichotomy between accessing content in DB and IR systems. In fact, recent trends in DB and IR research demonstrate a growing interest in adopting IR techniques in DBs and vice versa. The goal of this report is to issue new challenges to both communities, in particular, from an application, end-user, querying and system architecture perspectives.

2. PANEL OVERVIEW

The panel included established DB and IR experts. We first list the set of questions asked to the panelists. We then present the viewpoint of each panelist and a summary of the discussion.

2.1 Panel Questions

1) Which real-world applications require a tight DB-IR integration? Can most applications be addressed by storing unstructured data as uninterpreted columns in a relational DB system, and invoking an IR engine over unstructured data?

2) XML is being touted as the dominant and pervasive standard that integrates structured and unstructured data, and XML query languages such as XQuery Full-Text [59], attempt to support this. Can we still cobble together a solution using traditional DB and IR systems? Or do we need to rethink the fundamental data management system architecture?

3) Does it make sense to evaluate "imprecise" queries over structured data and produce ranked results? Conversely, does it make sense to evaluate "precise and complex" queries over unstructured or semi-structured data? If so, do any of the IR techniques carry over to the structured domain, and vice versa? Does this then argue for or against a unified query model?

4) DB and IR systems are already complex pieces of software with decades of research and a strong commercial backing. Is it possible to design a clean underlying formal model (akin to the relational model and IR ranking models) that captures the whole gamut of issues that both classes of systems deal with? Is it feasible to build a system based on what could be exceedingly complex data and query models? Would this gain acceptance in the marketplace and displace loosely coupled DB and IR systems?

5) Are there any "cultural" issues that would prevent a true DB-IR unification?

2.2 Panel Discussion

The panelists selection covered different perspectives of the panel topic. Pat Case, a librarian at the Congressional Research Service at the U.S. Library of Congress, gave her expert user's view on combining full-text search with structured search. Then, Gerhard Weikum, a research director at the Max-Planck Institute of Computer Science in Saarbruecken, Germany, presented an applications' perspective on the integration of DB and IR technologies. The following panelist, Thomas Rölleke, a research fellow and lecturer at Queen Mary University in London, provided an IR-expert view on the panel topic. Finally, Jayavel Shanmugasundaram, an assistant professor at the Department of Computer Science at Cornell University, described his system architecture's viewpoints.

Pat Case, the first panelist, motivated the need for a search system that integrates DB and IR querying capabilities. She stated the fact that existing solutions lack some fundamental features needed by expert users who need to search a database of documents, such as the document repository at the Library of Congress, as opposed to searching the open Web. The first requirement of a good system is the ability to return fewer results since end-users must be

[1] Panel slides available at:
www.research.att.com/sihem/SIGMOD-PANEL/.

able to review all of the results. A good search system must allow users to refine their search results by explicitly limiting or expanding the number of answers or by using taxonomies and ontologies. The second requirement is the ability to parameterize the scoring method used to rank query answers. Most IR engines [2] are treated as black boxes which use proprietary scoring algorithms to decide, on behalf of end-users, how to rank query results. Pat argued for relevance that is based on user-specified criteria, not on some word frequency method such as tf*idf. As an example, a congressional bill is more relevant if it is of a certain bill type, if it has been reported out of committee, placed on calendar, discussed on the floor, passed by one chamber, has become law, has a large number of co-sponsors etc. In addition, a system should also permit exact and unscored searches. The third requirement is the need for ordered and unordered word distance operators. With the advanced search functionality provided in today's search engines, users get OR (which is useless, except for strings of synonyms, AND which is close to useless, NOT, which is dangerous and, PHRASE which is way too limiting and it is a lie in some systems! More generally, a search system should offer a full array of full-text search functionalities. In January 2005, the PEW Internet & American Life Project released a report titled: "Internet Searchers are confident, satisfied and trusting, but they are also unaware and naive". It noted that only 7% of users use more than 3 search engines on a regular basis. However, these are librarians, researchers, doctors, lawyers, scientists, academics, and the graduate students who need to know everything that has been written on their dissertation topic. Example functionality that would help such users is prefix, infix, and suffix wild cards, ordered and unordered distance operators, thesaurus integration, starts-with functionality, a usable NOT, and end user control over diacritics, case, and stop words. In addition to powerful text search primitives, Pat argued for the necessity to combine them with a full array of SQL-like searches on dates, numbers, strings, and nodes. Examples of such queries are date and number range searching and the ability to search within a single instance of a field or element. Right know, librarians are forced to choose between full-text and SQL-like search functionalities. At the Library of Congress (LoC), document metadata is ported from a relational database to a full-text search engine. As a result, the SQL search capabilities are lost. Finally, Pat argued for a standard end user syntax that combines structured and unstructured search and that can be used reliably across search systems.

The next panelist, Gerhard Weikum, described a number of applications such as customer support and health care management. In both cases, text such as problem descriptions (in customer support) or symptoms (in health care management) are connected to structured data such as location and time. Such applications thus require queries on both text and data. Moreover, they usually require ranked result lists rather than result sets. So the IR paradigm of ranked retrieval, based on probabilistic models of relevance, should be carried over to the world of structured data, too, and further lead to a unified ranking methodology for all kinds of combined information. This becomes even more important in the context of data integration. These days many scientific and business applications need to combine and analyze data that comes from different sources. Ideally, this would require reconciling schemas, identifying and linking matching entities in the data instances, and cleaning and transforming values. However, this kind of data integration is almost always the bottleneck, and often users would be gladly willing to work with less perfect data, with statistically "guessed" or "learned" matchings and approximately cleaned-up data values. An example of a technique that helps integration but naturally introduces such uncertainty is entity recognition which combines natural language processing methods with pattern matching and Markov-model-based learning in order to extract persons, products, etc. from text. Global queries on data sources that are partially and approximately integrated using such statistical and heuristic techniques naturally require ranked retrieval. Gerhard finished his presentation with a number of recommendations including (i) work on Approximate Query Processing, Statistics-based Information Extraction, (ii) integrate logic-based and statistics-based paradigms and establish foundations for probabilistic SQL and XQuery, (iii) develop system architectures for flexible scoring and ranking, (iv) develop cognitive models of user intentions and behavior, (v) develop a better experimental methodology towards reproducible results and more objective insights into efficiency/quality tradeoffs and, (vi) think about an integrated DB&IR curriculum. Finally, in order to address the "cultural" barrier Gerhard suggested to colocate the SIGMOD and SIGIR conferences.

The next panelist, Thomas Rölleke, argued that while DB research focuses on relational data modeling, SQL and transaction-based processing, IR research focuses on text document retrieval. As a result, although new trends such as multimedia applications and querying XML document collections are a driving force for the integration of IR and DB approaches, integrating both technologies in the same system is not feasible. This is also due to the fact that technology used in today's IT environments comprises vertical solutions for DB, enterprise, web and document search rather than integrated technology. IR yields the methods for relevance-based ranking, while DB research provides methods for dealing with structured, and, increasingly, semi-structured data. The integration is technologically challenging, and the question is whether an IR application on top of classical SQL technology meets the requirements and scalability of IR applications. Thomas argues that changes in the relational algebra core (management of uncertainty, stream-based processing) are needed for meeting IR requirements. Also, the cultural integration of the research communities is actually even more challenging than the technological integration. Thomas was one of the organizers of a SIGIR 2004 workshop on integration of DB and IR. However, he believes that while a unified DB and IR system is needed to improve expressiveness, scalability and abstraction, and, overall, productivity [20], as far as XML applications are concerned, XML on top of new relational IR technology works fine in practice.

The last panelist, Jayavel Shanmugasundaram, presented three alternative approaches for unifying DB and IR and argued that the first two options do not work. The first approach, which ties together existing DB and IR systems such as the one taken by SQL/MM [41], is not powerful enough since both systems are treated as black boxes. The second approach is based on extending DB systems with IR functionality, or vice versa. Jayavel argued that extending (R)DBMSs violates many assumptions hardwired into current database systems. For example, is author name a structured or text field? In addition, database operators have precise, well-defined semantics while in IR, even the query result is not well-defined. In addition, scoring is databases is an attribute tacked on as a relational column and it is not clear how it can generalize IR scoring. Jayavel also argued that extending an IR system would not work because IR systems provide little support for structured data. In addition, scoring does not take structure into account. Finally, Jayavel argued for a new system architecture that would eventually replace today's systems and that is based on three design principles: (i) *structural data independence* which should guarantee that users

[2] http://www.lexisnexis.com,
http://www.google.com,
http://thomas.loc.gov

can issue complex and keyword queries over structured and unstructured data, (ii) *generalized scoring* that operates over any mix of structured and unstructured data (e.g., XRank over HTML and XML [31]) and, (iii) *a flexible and powerful query language* that allows for arbitrary return results and scores (e.g., TeXQuery [3], XQuery Full-Text [59] and NEXI [34] languages).

2.3 Summary

1. *Potential data and applications* include LoC documents available at:
   http://www.loc.gov, a LoC search engine at http://thomas.loc.gov and customer support and Health care management.

2. *Research ideas:* (1) Realizing IR functionality in a DB system, and vice versa, provides a limited integration of their functionalities but could be a good solution for some applications where the main focus is on one kind of data or the other; (2) Standard end-user syntax (see XQuery Full-Text for XML search [59] but how about for non-XML data formats?); (3) Generalized scoring on structured and text content; (4) Approximate SQL, top-K ranking, parameterized ranking; (5) Approximate data integration and data cleaning; (6) New system architecture to unify DB and IR.

3. *Organizational ideas* include co-locating SIGIR and SIGMOD and participating to the INEX [34] and W3C FTTF efforts [59].

3. BIOS OF PANEL PARTICIPANTS

3.1 Moderator

Sihem Amer-Yahia is a researcher at AT&T Labs Research. She received her Ph.D. in Computer Science from the University Paris XI-Orsay and INRIA. She has been working on various issues related to XML query processing. Sihem is a co-editor of the XQuery Full-Text language specification [59] and use cases [58] published in September 2005 by the Full-Text Task Force in the W3C whose charter is to extend XQuery with full-text search and ranking capabilities. She is currently involved in the GalaTex project (www.galaxquery.org/galatex), a conformance implementation of XQuery Full-Text.

3.2 Panelists

bf Pat Case works for the Congressional Research Service at the U.S. Library of Congress. She is a Librarian who works as a search interface designer for the Legislative Information System – the Congress-access-only version of thomas.loc.gov/. Pat is a co-editor of the XQuery Full-Text language specification [59] and use cases [58] published in April 2005 by the Full-Text Task Force in the W3C whose charter is to extend XQuery with full-text search and ranking capabilities.

Thomas Rölleke attended from 1984-1986 a private computer school of former Nixdorf Computer. From 1986-1988, he was a management trainee and product consultant in the Unix marketing of Nixdorf Computer. In 1988, he started his studies in Computer Science, and obtained his MSc in 1994. In 1999, he obtained his PhD on "POOL: A probabilistic object-oriented logic for information retrieval". Since 2000, he has been working as strategic IT consultant for a leading online-bank, company directory, research fellow and lecturer at Queen Mary University in London (QMUL). Thomas Rölleke is currently the director of QMUL's first computer science spin-out. He holds a patent for a new SQL variant to support relevance-based retrieval in relational DBs. His research and activities are shaped by the vision that the integration of modern IR and DB technologies is an important step for increasing the productivity in building advanced information systems.

Jayavel Shanmugasundaram is an Assistant Professor at the Department of Computer Science at Cornell University. He obtained his Ph.D. degree from the University of Wisconsin, Madison. Prior to joining Cornell University, he spent two years at the IBM Almaden Research Center in San Jose, California. Jayavel's research interests include Internet data management, IR, and query processing in emerging system architectures. He is an invited expert to the W3C Full-Text Task Force, and is also the recipient of the NSF CAREER Award and an IBM Faculty Award.

Gerhard Weikum is a Research Director at the Max-Planck Institute of Computer Science in Saarbruecken, Germany. Earlier affiliations include the University of the Saarland in Germany, ETH Zurich in Switzerland, MCC in Austin, Texas, and, during a sabbatical, Microsoft Research in Redmond, Washington. Gerhard is co-author of more than 100 refereed publications, and he has written a textbook on Transactional Information Systems, published by Morgan Kaufmann. He received the 2002 VLDB ten-year award for his work on automatic tuning. His current research interests include intelligent search on semistructured data, combining DB technology with IR techniques, and "autonomic" peer-to-peer information management. Gerhard serves on the editorial boards of ACM TODS and IEEE CS TKDE, and he was the program committee chair for the 2004 SIGMOD conference in Paris.

4. REFERENCES

[1] S. Agrawal, S. Chaudhuri, G. Das, A. Gionis: Automated Ranking of Database Query Results. CIDR 2003.

[2] S. Al-Khalifa, C. Yu, H.V. Jagadish. Querying Structured Text in an XML Database. SIGMOD 2003.

[3] S. Amer-Yahia, C. Botev, J. Shanmugasundaram. TeXQuery: A Full-Text Search Extension to XQuery. WWW 2004.

[4] S. Amer-Yahia, N. Koudas, A. Marian, D. Srivastava, D. Toman Structure and Content Scoring for XML. To appear in VLDB 2005.

[5] S. Amer-Yahia, L. Lakshmanan, S. Pandit. FleXPath: Flexible Structure and Full-Text Querying for XML. SIGMOD 2004.

[6] R. Baeza-Yates, B. Ribeiro-Neto. Modern Information Retrieval. Addison-Wesley, 1999.

[7] R.A. Baeza-Yates, M.P. Consens. The Continued Saga of DB-IR Integration. Tutorial, VLDB 2004.

[8] A. Balmin, V. Hristidis, Y. Papakonstantinou. ObjectRank: Authority-Based Keyword Search in Databases. VLDB 2004.

[9] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, S. Sudarshan. Keyword Searching and Browsing in Databases using BANKS. ICDE 2002.

[10] H.M. Blanken, T. Grabs, H.-J. Schek, R. Schenkel, G. Weikum (Editors). Intelligent Search on XML Data, Applications, Languages, Models, Implementations, and Benchmarks. Springer, 2003.

[11] J. Bosak. The plays of Shakespeare in XML. http://www.oasis-open.org/cover/bosakShakespeare200.html

[12] J. M. Bremer, M. Gertz. XQuery/IR: Integrating XML Document and Data Retrieval. WebDB 2002.

[13] C. Botev, J. Shanmugasundaram. Context-Sensitive Keyword Search and Ranking for XML. WebDB 2005.

[14] E. W. Brown. Fast Evaluation of Structured Queries for Information Retrieval. SIGIR 1995.

[15] D. Carmel, Y. S. Maarek, M. Mandelbrod, Y. Mass, A. Soffer. Searching XML Documents via XML Fragments. SIGIR 2003.
[16] Soumen Chakrabarti. Breaking Through the Syntax Barrier: Searching with Entities and Relations. ECML 2004 and PKDD 2004.
[17] S. Chaudhuri, G. Das, V. Hristidis, G. Weikum. Probabilistic Ranking of Database Query Results. VLDB 2004.
[18] S. Chaudhuri, R. Ramakrishnan, G. Weikum. Integrating DB and IR Technologies: What is the Sound of One Hand Clapping? CIDR 2005.
[19] T. T. Chinenyanga, N. Kushmerick. Expressive and Efficient Ranked Querying of XML Data. WebDB 2001.
[20] E. F. Codd. A Relational Model of Data for Large Shared Data Banks. Commun. ACM 13(6): 377-387 (1970).
[21] E.F. Codd. Relational Completeness of Database Sublanguages. In R. Rustin (ed.), Database Systems, Prentice-Hall, 1972.
[22] S. Cohen, J. Mamou, Y. Kanza, Y. Sagiv. XSEarch: A Semantic Search Engine for XML. VLDB 2003.
[23] W.W. Cohen. Integration of Heterogeneous Databases Without Common Domains Using Queries Based on Textual Similarity. SIGMOD 1998.
[24] W. B. Croft. Language Models for Information Retrieval. Invited Talk. ICDE 2003.
[25] D. Florescu, D. Kossmann, I. Manolescu. Integrating Keyword Search into XML Query Processing. WWW 2000.
[26] DBLP in XML. http://dblp.uni-trier.de/xml/
[27] N. Fuhr, K. Grossjohann. XIRQL: An Extension of XQL for Information Retrieval. SIGIR 2001.
[28] N. Fuhr, K. Grossjohann. XIRQL: An XML query language based on information retrieval concepts. ACM TOIS 22(2), 2004.
[29] N. Fuhr, T. Rölleke. A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems. ACM TOIS 15(1), 1997.
[30] T. Grabs, K. Böhm, H.-J. Schek. PowerDB-IR - Information Retrieval on Top of a Database Cluster. CIKM 2001.
[31] L. Guo, F. Shao, C. Botev, J. Shanmugasundaram. XRANK: Ranked Keyword Search over XML Documents. SIGMOD 2003.
[32] Health Level Seven. http://www.hl7.org
[33] V. Hristidis, L. Gravano, Y. Papakonstantinou. Efficient IR-Style Keyword Search over Relational Databases. VLDB 2003.
[34] Initiative for the Evaluation of XML Retrieval. http://inex.is.informatik.uni-duisburg.de:2004/
[35] V. Kakade, P. Raghavan. Encoding XML in Vector Spaces. ECIR 2005.
[36] R. Kaushik, R. Krishnamurthy, J.F. Naughton, R. Ramakrishnan. On the Integration of Structure Indexes and Inverted Lists. SIGMOD 2004.
[37] B. Kimelfeld,Y. Sagiv. Efficient Engines for Keyword Proximity Search. WebDB 2005.
[38] M. Lalmas, T. Rölleke. Modelling Vague Content and Structure Querying in XML Retrieval with a Probabilistic Object-Relational Framework. FQAS 2004.
[39] Library of Congress. http://lcweb.loc.gov/crsinfo/xml/
[40] S. Liu, R. Shahinian, W. Chu. XML Vague Content and Structure (VCAS) Retrieval over Document-centric XML Collections. WebDB2005.
[41] J. Melton, A. Eisenberg. SQL Multimedia and Application Packages (SQL/MM). SIGMOD Record 30(4), 2001.
[42] A. Marian, S. Amer-Yahia, N. Koudas, D. Srivastava. Adaptive Processing of Top-k Queries in XML. ICDE 2005.
[43] J. Naughton, et al. The Niagara Internet Query System. IEEE Data Engineering Bulletin 24(2), 2001.
[44] N. Polyzotis, M. Garofalakis, Y. Ioannidis. Approximate XML Query Answers. SIGMOD 2004.
[45] S. Robertson. The probability ranking principle in IR. Journal of Documentation 33, 1977.
[46] A. Salminen. A Relational Model for Unstructured Documents. SIGIR 1987.
[47] G. Salton, M. J. McGill. Introduction to Modern Information Retrieval. McGraw-Hill, 1983.
[48] G. Salton, and A. Wong. A Vector Space Model for Automatic Indexing. Communications of the ACM 18, 1975.
[49] D. Shin, H. Jang, H. Jin. BUS: An Effective Indexing and Retrieval Scheme in Structured Documents. Proc. 3rd Int. Conf. on Dig. Lib., 1998.
[50] T. Schlieder. Schema-Driven Evaluation of Approximate Tree-Pattern Queries. EDBT 2002.
[51] A. Theobald, G. Weikum. Adding Relevance to XML. WebDB 2000.
[52] A. Theobald, G. Weikum. The Index-Based XXL Search Engine for Querying XML Data with Relevance Ranking. EDBT 2002.
[53] M. Theobald, R. Schenkel, G. Weikum. An Efficient and Versatile Query Engine for TopX Search. To appear in VLDB 2005.
[54] H. Turtle, B. Croft. Inference Networks for Document Retrieval. SIGIR 1990.
[55] F. Weigel, H. Meuss, K. U. Schulz, F. Bry. Content and Structure in Indexing and Ranking XML. WebDB 2004.
[56] The World Wide Web Consortium. XQuery 1.0: An XML Query Language. W3C Working Draft. http://www.w3.org/TR/xquery/.
[57] The World Wide Web Consortium. XML Path Language (XPath) 2.0. W3C Working Draft. http://www.w3.org/TR/xpath20/
[58] The World Wide Web Consortium. XQuery 1.0 and XPath 2.0 Full-Text Use Cases. W3C Working Draft. http://www.w3.org/TR/xmlquery-full-text-use-cases/
[59] The World Wide Web Consortium. XQuery 1.0 and XPath 2.0 Full-Text. W3C Working Draft. http://www.w3.org/TR/xquery-full-text/
[60] The World Wide Web Consortium. XQuery 1.0 and XPath 2.0 Functions and Operators. W3C Working Draft. http://www.w3.org/TR/xquery-operators/
[61] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, G. Lohman. On Supporting Containment Queries in Relational Database Management Systems. SIGMOD 2001.
[62] E. Zimanyi. Query Evaluations in Probabilistic Relational Databases. Theoretical Computer Science, 1997.

Report on the First IEEE International Workshop on Networking Meets Databases (NetDB'05)

Cyrus Shahabi, Ramesh Govindan
University of Southern California
Department of Computer Science
Los Angeles, CA 90089-0781
U.S.A.
[shahabi , ramesh]@usc.edu

and

Karl Aberer
EPFL
Computer & Communication Science
Bâtiment BC, Station 14
CH-1015 Lausanne, Switzerland
karl.aberer@epfl.ch

In this report, to the best of our ability, we try to summarize the presentations and discussions occurred within the First IEEE International Workshop on Networking Meets Databases (NetDB) which was held in Tokyo Japan on April 8th and 9th, 2005. NetDB was one of the many (11 to be exact) satellite workshops of the IEEE ICDE (International Conference on Data Engineering) 2005 conference. This workshop is part of the very few initiatives in bringing the networking and database communities together. The focus research areas of NetDB 2005 were sensor and peer-to-peer networks.

1. Objectives and Justifications

The objective of NetDB is to bring together researchers in the networking and database communities to debate emerging research directions at the intersection of these two fields. We are witnessing the blurring of the traditional boundaries between these two disciplines, especially in the emerging areas of sensor and peer-to-peer networks. We believe the time is ripe for these two communities to get together and discuss the common interests, share and exchange expertise and results, and appreciate each other's terminologies and contributions. The goal of this year's workshop was to promote discussion of ideas that will influence and foster continued research in the areas of sensor and peer-to-peer networks. The workshop provides a venue for researchers to present new ideas that can significantly impact both communities and perhaps give birth to a new community in the long term.

The main reason we selected peer-to-peer and sensor networks as the focused areas for this year's workshop was due to growing number of research papers in these two areas authored by both the database and networking communities. More importantly, unlike other target applications, the research papers appearing in each community's forums are not always addressing complementary research issues but mostly the same exact problems, sometimes following the same approaches. Hence, the question was whether it makes sense to have joint forums, such as NetDB, to just consolidate terminologies and encourage closer interactions. And the broader question is if a new area is emerging such as what happened when some of the database, signal processing and computer vision researches interacted and gave rise to the "multimedia" community or whether this is just a temporary trend that will pass.

2. Logistics and Statistics

The workshop was spread into two half-days in the afternoon of April 8th and the morning of the April 9th. We had an impressive program committee with 17 expert members from both areas of networking and databases. They professionally reviewed seventeen papers (at least 3 reviews per submission), out of which nine papers were accepted for presentation in the workshop.

About thirty people registered for the workshop that given the date of the workshop (the last day of the conference), the competition (all the other workshops running in parallel) and the number of accepted papers (nine) was very encouraging and to be honest pleasantly surprising. We think this should partly be attributed to our excellent keynote speaker (Mike Franklin) and to our celebrity panel members (Karl Aberer, Amr El Abbadi, Ramesh Jain, Dimitrios Gunopulos and Wei Hong). Also in the audience, in addition to the chairs and panel members, we had the following PC members: Gustavo Alonso and Ugur Cetintemel. Our special thanks to all of them as well as to the authors, the rest of the PC and the audience for making this first NetDB workshop a success. Finally, our gratitude goes to Prof. Masaru Kitsuregawa for his unlimited support for this workshop.

3. Technical Presentations

The nine paper presentations were distributed into four sessions, two sessions on April 8th and two sessions on April 9th. The papers' topics were very much consistent with the ideas behind the workshop's objectives and interestingly in a very balanced way. Consequently, we evenly distributed the papers between the following four sessions: networking, databases, peer-to-peer networks and sensor networks. The entire program including links to the papers is available at: http://infolab.usc.edu/netdb05/ .

The focus of the papers under the **database session** was mainly on query processing issues. The first paper of this session, by a group of authors from Harvard University, discusses ways to optimize queries for distributed stream-based applications where the network is large in scale and dynamic. The second paper, from Brown University, focuses on evaluating the database join operator in heterogeneous networks with the objective of expediting the output of the result.

The papers under the **networking session** considered the application of database techniques to network measurement and monitoring. In the first paper in this session, Li et al. argue that wide area network monitoring can be naturally supported by a distributed indexing system that enables multidimensional range queries. The second paper in the session pertains to stream processing, and argues that relatively simple information theoretic approaches work well in helping to detect qualitative changes in the Internet traffic streams.

The **peer-to-peer networks session** was dedicated to two papers focused on the data access issues with DHT-based peer-to-peer databases. The first paper, which is a result of collaboration between researchers from two German universities (TU Ilmenau and UniMagdeburg), promotes query approximation as a solution for he problem of uncertain data availability in dynamic peer-to-peer databases. Particularly, this paper defines approximation semantics for aggregate queries and similarity queries, and introduces two query evaluations methods to answer such approximate queries. The second paper, from EPFL, considers the problem of non-uniform data distribution in peer-to-peer data networks, and inspired by small-world models proposes a generalization for the DHT family that preserves the efficiency of DHT-based data access even under skewed data distribution scenarios.

Finally, the **sensor networks session** examined the application of database techniques to networked sensing. The energy and storage constraints in this domain give rise to novel approaches. Two of the papers consider the energy efficient processing of historical queries; Coman et al. analytically examine the efficacy of a variety of query routing strategies, while Zeinalipour-Yazti et al. discuss the query processing opportunities and storage management challenges presented by low-power sensing platforms having several megabytes of storage. The other paper in this session argues that sensor network data must be named by its *provenance*--- the origin of the data and the history of operations on it. Such an approach leads to novel data management and query processing issues, and the paper delineates them well.

4. Panel Discussion and Keynote Talk

The panel was the last event of the first day of the workshop in order to keep the audience around until the last minute! The topic of the panel, *"Networking Meets Databases: Do we meet or merge?"* was one of the main questions we wanted to discuss in this workshop. To be specific, whether it is sufficient for these two communities to have occasional joint conferences such as NetDB or a new community should emerge addressing the common challenges?

This is both a technical question, whether technical problems exist that require joint research efforts by both communities, and a cultural question, in how far the existing communities with their different habits, terminologies and backgrounds are fit to interact.

Panel statements were given by Amr el Abbadi, Wei Hong, Dimitrios Gunopolous and Ramesh Jain. All four panelists were of the opinion that sensor networks are one of the driving forces of bringing the communities closer together. Wei Hong claimed that convergence is already here and that it is an important development to study important questions such as layering vs. cross-layer optimization, declarativeness vs. expressiveness and performance vs. predictability. Dimitrios Gunopolous emphasized the role of P2P as a powerful model for developing infrastructure-less Internet-Scale systems. For example P2P could overcome many problems of current search engines. Interdisciplinary research is the way to go there, but the fields have to keep their core strength. Amr el Abbadi pointed out that content-based addressing is already a common concern. However, he observed that conferences of each community have their own idiosyncrasies, which prevent cross-fertilization. Ramesh Jain elaborated on his vision of a document and an event Web. This development will generate many new research problems which in his opinion are neither solely addressable by the networking nor database communities.

In the subsequent discussion, the issue of publishing across communities was hotly debated. It became clear that there are cultural problems to be overcome, but also that the trend is towards opening, as shown by many concrete examples of cross-fertilization and cross-publishing.

The second day of the workshop started with the keynote talk (again to bring the audience to the workshop from the first minute!) delivered by Mike Franklin. In his talk, Mike defined the concept of *high fan-in architectures* for large scale applications where vast numbers of events measured at the edges of the network are continually refined, summarized, augmented, and aggregated as they flow towards the interior. Subsequently, he discussed the key characteristics and challenges presented by high fan-in systems, and argued for a uniform, query-based approach towards addressing them.

Some of discussions following Mike's talk were more on the differences between peer-to-peer and sensor networks. Some in the audience believed that there are more funding opportunities for sensor network projects while projects in peer-to-peer networks usually get labeled with illegal music/video sharing and hence has a negative spin associated with them. Otherwise, it seemed that the audience agrees that both applications share some common underlying challenges.

5. Where we go from here?

Obviously, many of the questions raised by this first workshop have not been addressed. Most importantly, it is not clear whether a new community is emerging or not. However, one clear conclusion is that the workshop should continue because of its huge success in its first year. The importance of the overlapping areas of networking and databases is very clear and the research in this overlapping area is healthy, active and well-funded.

Encouraged by the success of the first workshop, we have created a steering committee for NetDB constituting of: Hari Balakrishnan (MIT), Michael Franklin (UC Berkeley), Ramesh Govindan (USC) and Cyrus Shahabi (USC). Obviously, the steering committee consists of two networking and two database researchers. In addition, following the first workshop, the 2nd NetDB workshop will be held with ICDE 2006 at Atlanta, GA. It also seems to become a tradition that the workshop be co-chaired by a networking and a database researcher. Hence, NetDB'06 is co-chaired by Ugur Cetintemel and John Jannotti from Brown University. The NetDB'06 website is at:
http://www.cs.brown.edu/research/db/netdb06/

XQuery 1.0 is Nearing Completion

Andrew Eisenberg
IBM, Westford, MA 01886
andrew.eisenberg@us.ibm.com

Jim Melton
Oracle Corp., Sandy, UT 84093
jim.melton@acm.org

Introduction

XQuery is a query language designed for querying real and virtual XML documents and collections of these documents. Its development began in the second half of 1999. We provided an early look at XQuery in Dec. 2002 [1]. XQuery 1.0 is now approaching its publication as a W3C Recommendation, and we would like to update you on its progress. We can speak to this area with even more authority than we did last time, as we both became co-chairs of the W3C XML Query Working Group [2] in summer 2004.

Paul Cotton (Microsoft), who chaired the group since its inception, stepped down from this role in October. His role in other consortia didn't allow him to stay with XQuery 1.0 all the way though its publication as a Recommendation, although he certainly wanted to do so. Paul deserves a great deal of credit for the leading role that he has played in the development of XQuery.

In his article, we concentrate on the changes that have taken place to XQuery since our earlier article. If you are unfamiliar with XQuery, then you may want to take a look at our earlier article before proceeding.

XQuery Status

The following documents [3] became Candidate Recommendations (CR) in November 2005.

- XQuery 1.0: An XML Query Language
- XML Path Language (XPath) 2.0
- XSL Transformations (XSLT) Version 2.0
- XQuery 1.0 and XPath 2.0 Data Model (XDM)
- XQuery 1.0 and XPath 2.0 Functions and Operators
- XQuery 1.0 and XPath 2.0 Formal Semantics
- XSLT 2.0 and XQuery 1.0 Serialization
- XML Syntax for XQuery 1.0 (XQueryX)

The XML Query WG has worked closely with the XSL WG on most of these documents.

Most of the documents underwent two Last Call Working Draft (WD) reviews, and a couple of them underwent three such reviews. In the last review, the WGs responded to approximately 600 comments.

The purpose of CR is to gain implementation experience and give a WG confidence that its specification is complete and unambiguous. To this end, the XML Query WG began the development of a test suite in the summer of 2004. The XML Query Test Suite [4] now covers about 75% of the features that make up XQuery. It will take several months for this test suite to be completed and to get reports back from implementers.

With luck, and some hard work on the part of a number of people, XQuery will become a W3C Recommendation before the end of 2006.

The XQuery 1.0 and XPath 2.0 Data Model (XDM)

XDM defines five types beyond those defined in XML Schema Part 2 [5]. Two of them were discussed in our previous article: `xdt:dayTimeDuration` and `xdt:yearMonthDuration`, where `xdt` is a prefix for the namespace `http://www.w3.org/2005/xpath-datatypes`.

`xdt:untyped` is assigned to element nodes that have not been validated or have been validated in skip mode. `xdt:untyped` is also the type assigned to a constructed element when the construction mode is `strip` (discussed later). All of the children of an element that is annotated as `xdt:untyped` are annotated as `xdt:untyped` as well.

`xdt:untypedAtomic` is assigned to values that are atomic, but which do not have a more specific type. An attribute that has been validated in skip mode are assigned this type.

`xdt:anyAtomicType` has `xs:anySimpleType` as its base type, and is the type from which all primitive atomic types are derived. These include types such as `xs:string`, `xs:float`, and `xdt:untypedAtomic`. This type is abstract in nature, as no values will be annotated with this type. From XQuery's point of view, this type has been inserted into the XML Schema type hierarchy.

Serialization

Since we wrote our earlier article, the WGs have created a new document, XSLT 2.0 and XQuery 1.0 Serialization [6]. The material in this document was removed from XSLT 2.0, so that it could be shared by XQuery 1.0. This document defines the XML, XHTML, HTML, and TEXT output methods. XQuery 1.0 makes use of only the XML output method, while XSLT uses all of them.

A value of the XQuery 1.0 and XPath 2.0 Data Model (XDM) may be provided for serialization. Sequence Normalization is performed, followed by markup generation, character expansion, and encoding.

Sequence Normalization is defined in several steps, transforming a data model instance—a sequence of values and nodes—into a single document node. Atomic values are cast into strings and then into text nodes. Document nodes are discarded, but their children are retained. It is a serialization error if an attribute node that is not a child of an element node is placed into the resulting document.

Serialization defines a number of parameters that influence the result that is produced. `omit-xml-declaration`, for example, can have either `yes` or `no` for its value. Not all parameters are used by a given output method.

The XML output method generates a well-form XML document entity if the result of sequence normalization is a document node with a single element node child and no text node children. Otherwise, a well-formed XML external general parsed entity is generated. The specification doesn't say how to form these entities. Instead, it requires that the same data model instance be produced by parsing the result and using the resulting infoset to generate a data model instance. Well, not exactly the same: it describes ways in which they are allowed to differ, such as the order in which attribute nodes appear.

No attempt is made to preserve the type annotations during serialization. If the result is XML Schema validated, then new type annotations will be created.

XQuery

XQuery 1.0 is almost a proper superset of XPath 2.0—XQuery 1.0 does not use XPath's namespace nodes and does not support XPath's namespace axis.

Inputs to XQuery Processing

The data model instances that XQuery can operate on can be provided in a number of ways. Our earlier article described the context item, denoted by ".", and the `fn:doc` and `fn:collection` functions. The `xf:input` function that we described earlier has been dropped in favor of external variables.

A variant of the `fn:collection` function without an argument has been introduced to refer to a default collection that may be supplied by the host environment.

Variables may be provided by an implementation for use in a query. A query may also define external variables and expect values for these variables to be provided by the host environment. The variable declaration may include a type for the variable. If it does not, then the host environment provides the variable's type as well as its value.

The following query might be executed with the `$custName` variable bound to "Big Box".

```
declare variable $custName as xs:string
external;

fn:doc('orders.xml')
   /orders/order[@cust=$custName]

→
<order id='444378' cust='Big Box'>
   ...
</order>
```

Steps that Return Atomic Values

In XPath 1.0, the result of a step in a path expression was a sequence of nodes in document order with duplicates removed. XQuery 1.0 and XPath 2.0 allow the final step in a path expression to produce a sequence of atomic values. A query for the cities and states of all California employees can be written as:

```
//employee[address/state='CA']
  /address/concat(city, ', ', state)
```

rather than:

```
for $a in //employee[address/state="CA"]
          /address
return concat($a/city, ', ', $a/state)
```

Declarations in the XQuery Prolog

A number of declarations have been added to XQuery's prolog. Some of these are the boundary-space declaration, base URI declaration, construction declaration, copy namespaces declaration, and option declaration. We'll discuss a couple of these in this section.

Boundary Space

The `boundary-space` declaration has values of `preserve` and `strip`, and determines whether boundary whitespace is preserved by element constructors. Let's look at an example:

```
declare boundary-space preserve;

<test>   <inner-element/>   </test>
```

The test element that is returned has 3 children; a text node containing several spaces, an element node, and another text node. If `strip` had been chosen, then the element node would be the only child. If this declaration is not used, then `strip` is the default.

Construction

The `construction` declaration also has values of `preserve` and `strip`. Here, a user chooses whether type annotations are preserved in the construction of new element and document nodes. If `strip` is chosen, then the constructed element node and all of its children are annotated with `xdt:untyped`, and all of its attribute nodes are annotated with `xdt:untypedAtomic`. If `preserve` is chosen, then the constructed element node is annotated with `xdt:anyType`, and all of its element nodes and attribute nodes retain their existing annotations.

Option

An `option` declaration is one of several extension mechanisms that XQuery provides to implementers. An option declaration contains a QName and string. If the QName is recognized by an implementation, then it can have whatever effect on the processing of the query the implementer chooses. If it is not recognized, then it is ignored. In this way, the extensions of one implementation will not cause execution on another implementation to fail.

Let's consider an extension that allows a user to set a timeout value, in seconds, after which the query will stop and return an error.

```
declare namespace myxquery='...';
declare option myxquery:timeout '10';

for $e in //employees ...
```

Expressions

`castable`, `extension`, `ordered`, and `unordered` have been added to the set of XQuery expressions and the syntax has been changed just a bit for `cast`, node comparison, and `validate`.

expression type	expression syntax
cast	*expr* `cast as` *type*
castable	*expr* `castable as` *type*

validate	`validate {` *expr* `}`
	`validate lax {` *expr* `}`
	`validate strict {` *expr* `}`

node comparison	`is` *(isnot was dropped)*
extension	*(see below)*
ordered	`ordered {` *expr* `}`
unordered	`unordered {` *expr* `}`

Castable

`castable` returns a Boolean value that indicates whether the value provided can be successfully cast to the type provided. Without this expression, a user would not be able to prevent the failure of a cast becoming a failure of the entire query. (Exception handling is something that might be considered in a future version of XQuery.)

Ordered and Unordered

An `ordered` expression sets the ordering mode to `ordered` for the expression that it contains. An `unordered` expression sets the ordering mode to `unordered`.

Path expressions that include a "`/`" or "`//`" operator or a step, set expressions (`union`, `intersect`, and `except`), and FLWOR expressions without an `order by` clause are sensitive to the setting of the ordering mode. When it is `ordered`, each produces its sequence of items in document order. When it is `unordered`, each produces its sequence of items in an arbitrary order. Relaxing the order of the items may allow an optimizer to choose a lower-cost strategy for evaluating the query.

The initial ordering mode can be set by a user in the query prolog. If it is not set, then the default ordering mode is `ordered`.

The following query returns New York employees in an arbitrary order, but it uses ordering in the inner path expression to select employees whose last title is "VP".

```
declare ordering unordered;

for $e in ordered {
  //employee[titles/title[last()] = 'VP'] }
where $e[location/@state='NY']
return $e
```

Validate

The `validate` expression applies XML Schema validation to its argument. Its argument is first converted into an infoset, discarding any type

annotations that it might have contained. The result of validation is a new element (with new contents and new identity) with type annotations. If validation is not successful, then a dynamic type error is raised.

Type annotations can be applied to a constructed element using the validate expression:

```
validate { <myco:employee id='440612'>
             <name>Augustus Child</name>
             .
             .
             .
           </myco:employee>
         }
```

In this case, the `myco` schema must contain a globally defined element `employee`. The `name` element in the constructed element has type `xdt:untyped`, while in the validated result it might have type `myco:nameType`.

Extension

An extension expression is another extension mechanism provided to implementers by XQuery. Where an `option` declaration has an effect for the entire query, an extension expression has a narrower scope. Let's use the following example to explain this construct.

```
declare namespace xq1="...";
declare namespace xq2="...";

for $e in //employee[name='Jon Postel']
return (# xq1:prose English #)
       (# xq2:roman lower-case #)
       { $e/badge cast as xs:string }
```

These *pragmas* "`(# … #)`", if they are recognized, might change the behavior of casting values to strings. This query might produce "One Hundred Fifty Four" if it recognizes `xq1:prose`, "cliv" if it recognizes `xq2:roman`, and "154" if it recognizes neither of them. The expression in curly braces "`{}`" can be omitted. If it is omitted and none of the pragmas is recognized, then an error is raised.

URI Values

XQuery has long allowed the type promotion of numeric values, from `xs:decimal` to `xs:float` and from `xs:float` to `xs:double`. Since our earlier article, XQuery has added promotion from `xs:anyURI` to `xs:string`.

Without this change, a query on an untyped document written as:

```
let $xq := 'http://www.w3.org/TR/xquery/'
return count(//bib[ref=$xq])
```

would cause a type error for a typed document due to the comparison of an `xs:anyURI` and an `xs:string` value. It would have to be rewritten as:

```
let $xq := 'http://www.w3.org/TR/xquery/'
return count(//bib[ref=xs:anyURI($xq)])
```

Types

Some of the type designators have changed since our last article. Rather than going through BNF, we'll just look a number of examples:

`xs:integer?`	a sequence of zero or one integer
`element()+`	a sequence of one or more elements
`node()*`	a sequence of zero or more nodes
`item()+`	one or more items
`attribute()`	an attribute (single) of any name and type
`element (myco:address)`	an element with name `myco:address`
`element (*, myco:addrType)`	an element of any name, with type `myco:addrType`
`schema-element(zip)`	an element named `zip` (or in a substitution group headed by `zip`) with a type annotation that matches the type of `zip` element

A type designator might be used as follows:

```
//employee
  [* instance of
     element (*, myco:addrType)

  ]
```

Earlier versions of XQuery allowed reference to be made to element and attributes that were locally declared in a schema, but this feature was dropped.

FLWOR Expression

The FLWR (for, let, where, return) expression has become the FLWOR expression (where "O" stands for "order by".

Each `order by` clause can contain multiple sort keys, each of which contains an expression and may contain an indication of whether the sorting should be stable, whether it should be ascending or descending, whether an empty sequence is considered greater than or less than any item, and whether a

collation sequence other than the default collation sequence should be used.

Each expression in the `order by` clause is evaluated for each of the bindings of the variables in the `for` and `let` clauses that are not eliminated by the `where` clause. If any expression produces a sequence of more than one item, then an error is returned. Any values that are of type `xdt:untypedAtomic` are cast to `xs:string`. If, for any sort key, the values differ in type (after considering subtype substitution and type promotion), then an error is returned.

The following query returns recently hired employees ordered first by their years of education and then by their department.

```
for $e in doc('employees.xml')//employee
where current-date() - $e/hireDate
      < xdt:dayTimeDuration('P60D')
order by
   $e/HSYears + $e/CollegeYears descending,
   $e/dept empty greatest
return $e
```

The choice of whether an empty sequence is greater than or less than an item can be made in the query prolog. An XQuery implementation can choose either of these as its default behavior.

The FLWOR expression also gained an `at` clause that binds the position of the item in the sequence at the same time that it binds the value of that item.

The following query returns the 10 employees that have been with the company the longest:

```
for $e at $p in
   (for $oe in //employee
    order by $oe/@hireDate descending
    return $oe)
where $p <= 10
return $e
```

In-scope Namespaces

XQuery has chosen not to support namespace nodes and a namespace axis, as XPath 1.0 did. Instead, XQuery associates a set of in-scope namespace bindings with its nodes.

XQuery also has a set of statically known namespaces, which are used when resolving its QNames. These statically known namespaces include `fn`, `xml`, `xs`, `xsi`, `xdt`, and `local`. An implementation may add its own namespace bindings, and a user may add to all of these bindings in the query prolog:

```
declare namespace
        myco="http://www.example.com/myco";

<myco:result> { for ... } </myco:result>
```

The in-scope namespaces may affect how an element node is serialized and may also affect the behavior of a small number of functions. The node constructed in this example has one namespace binding associated with it. The namespace for `myco` is taken from the statically known namespaces when the node is constructed.

When a node is constructed, its namespace bindings include the one used in the element name, those used in the attribute names, those defined by namespace declaration attributes, and those in namespace attribute declarations of enclosing element constructors that have not been overwritten.

Let's consider the following example:

```
import schema namespace hr="...";

validate strict {
   <hr:employee>
      <hr:skill xsi:type="xs:string">
         unicycling
      </hr:skill>
   </hr:employee>
}
```

This query will raise an error, because a binding for `xs` will not appear in the infoset that is validated. The `xsi:type` attribute is given no special consideration by XQuery. "xs:string" is just an untyped attribute value, it is not seen as a QName, and so `xs` does not get added to the in-scope namespaces. This means that it does not become part of the infoset. This query can be fixed by changing the start tag as follows:

```
<hr:employee xmlns:xs
      ="http://www.w3.org/2001/XMLSchema">
   .
   .
   .
</hr:employee>
```

Finer-grained control over the in-scope namespaces of constructed nodes is available to a user via the `copy-namespaces` declaration in the query prolog.

Modules

A library module is a collection of variables and functions in a target namespace that can be imported into a query.

```
module namespace univ
       ="http://www.example.com/university";

declare function univ:gpa
   ($e as element (student)) as xs:decimal
   { for ... } ;
```

This function could be invoked in a query in the following way:

```
import module namespace univ
       ="http://www.example.com/university";
declare variable $id external;

univ:gpa(//student[id=$id])
```

XQueryX

XQueryX [9] defines an XML representation of XQuery. It defines an element structure that mirrors the abstract syntax of XQuery. The definition of XQueryX has changed quite a bit since we showed it to you last. Example 1 contains a simple XQuery and the corresponding XQueryX representation.

While XQueryX is harder for a human to read and write than XQuery, it does have several useful properties. It is easily generated by tools and layered applications, it is easily embedded within larger XML documents, and it allows "queries on queries".

Of course, all changes made to XQuery apply equally to XQueryX. But there is another fairly important change that has been made to XQueryX. When we last showed it to you, the XML Schema that defines the XQueryX syntax was based on a sort of type hierarchy that turned out to be difficult to maintain as new features were added to the language, and also somewhat difficult for human readers to keep in their minds. That hierarchical design has been replaced with one based on XML Schema's substitution groups. This sort of approach is more readily extensible when new language features are created, and also more familiar to Schema experts.

```
for $b in .//book
return $b/title
→
<xqx:module
   xmlns:xqx="http://www.w3.org/2005/XQueryX"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <xqx:mainModule>
    <xqx:queryBody>
      <xqx:flworExpr>
        <xqx:forClause>
          <xqx:forClauseItem>
            <xqx:typedVariableBinding>
              <xqx:varName>b</xqx:varName>
            </xqx:typedVariableBinding>
            <xqx:forExpr>
              <xqx:pathExpr>
                <xqx:argExpr>
                  <xqx:contextItemExpr/>
                </xqx:argExpr>
                <xqx:stepExpr>
                  <xqx:xpathAxis>
                     descendant-or-self
                  </xqx:xpathAxis>
                  <xqx:anyKindTest/>
                </xqx:stepExpr>
                <xqx:stepExpr>
                  <xqx:xpathAxis>child</xqx:xpathAxis>
                  <xqx:nameTest>book</xqx:nameTest>
                </xqx:stepExpr>
              </xqx:pathExpr>
            </xqx:forExpr>
          </xqx:forClauseItem>
        </xqx:forClause>
        <xqx:returnClause>
            .
            .
            .
        </xqx:returnClause>
      </xqx:flworExpr>
    </xqx:queryBody>
  </xqx:mainModule>
</xqx:module>
```

Example 1 – Equivalent XQuery and XQueryX

Several minor changes were also made to XQueryX's Schema. The most significant change is a new Schema element, <xqx:xquery>, used for a trivial embedding of XQuery (human-readable) text into XML documents.

XQuery and XQueryX Conformance

Both XQuery and XQueryX have conformance statements that define Minimal Conformance and a set of optional features.

Minimal Conformance is the lowest level of conformance that can be claimed for XQuery. Minimal Conformance encompasses all XQuery functionality, with the exception of the following optional features:

- Schema Import Feature – allow the use of `import schema` in the prolog to make XQuery aware of the declarations of elements, attributes, and types.
- Schema Validation Feature – allows the use of the `validate` expression.
- Static Typing Feature – requires XQuery to detect and report type errors during the static analysis phase. Some queries that might run successfully without static typing will return an error during static analysis.
- Full Axis Feature – allows the use of the "reverse axes" `ancestor`, `ancestor-or-self`, `following`, `following-sibling`, `preceding`, and `preceding-sibling`.
- Module Feature – allows the use of `import module` in the prolog and allows library modules to be created.
- Serialization Feature – requires that an implementation provide a way to produce an XML serialization of the result of a query.
- Trivial XML Embedding Feature – allows an query to be provided as an XML element.

```
<xqx:xquery>for $e in ... </xqx:xquery>
```

Future Work

While we continue to move XQuery 1.0 through the W3C process towards its publication as a Recommendation, we have work underway that will add to XQuery 1.0.

Several Working Drafts (WD) have been published for XQuery 1.0 and Path 2.0 Full-Text [7]. Requirements have been published for an XQuery Update Facility [8], but an initial WD has not yet been published.

We expect that early next year the XML Query WG will begin considering features that could not be included in XQuery 1.0 for a future version of this Recommendation.

References

[1] *An Early Look at XQuery*, Andrew Eisenberg and Jim Melton, ACM SIGMOD Record, Vol. 31, No. 4, December 2002, http://www.sigmod.org/sigmod/record/issues/0212/AndrewEJimM.pdf.

[2] W3C XML Query (XQuery), http://www.w3.org/XML/Query/.

[3] W3C Technical Reports and Publications, http://www.w3.org/TR/.

[4] XML Query Test Suite, http://www.w3.org/XML/Query/test-suite/.

[5] *XML Schema Part 2: Datatypes Second Edition*, Paul V. Biron and Ashok Malhotra, Oct. 28, 2004, http://www.w3.org/TR/xmlschema-2/.

[6] *XSLT 2.0 and XQuery 1.0 Serialization*, Michael Kay, et al, Nov. 3, 2005, http://www.w3.org/TR/xslt-xquery-serialization/.

[7] *XQuery 1.0 and XPath 2.0 Full-Text*, Sihem Amer-Yahia, et al, Nov. 3, 2005, http://www.w3.org/TR/xquery-full-text/.

[8] *XQuery Update Facility Requirements*, Don Chamberlin and Jonathan Robie, June 3, 2005, http://www.w3.org/TR/xquery-update-requirements/.

[9] *XML Syntax for XQuery1.0*, Jim Melton and Subramanian Muralidhar, Nov. 3, 2005, http://www.w3.org/TR/xqueryx/.

Christos Faloutsos Speaks Out
on Power Laws, Fractals, the Future of Data Mining, Sabbaticals, and More

by Marianne Winslett

Christos Faloutsos
http://www.cs.cmu.edu/~christos/

Welcome to this installment of ACM SIGMOD Record's series of interviews with distinguished members of the database community. I'm Marianne Winslett, and today I have here with me Christos Faloutsos, who is a professor of computer science at Carnegie Mellon University. Christos recieved the Presidential Young Investigator Award from the National Science Foundation in 1989. He received the 1997 VLDB Ten Year Paper Award for his paper on R+ trees, and the SIGMOD 1994 Best Paper Award for a paper on fast subsequence matching in time series databases. Christos is a member of the SIGKDD Executive Committee, and he has wide-ranging interests in data mining, database performance, and spatial and multimedia databases. His PhD is from the University of Toronto. So, Christos, welcome!

Thank you very much, Marianne, for having me here.

Christos, you have been described as the master of collaborations: someone who can reach out to colleagues not only in the database area, but also in other disciplines, such as the sciences and in statistics, and to visitors from industry as well. What do you do to nurture such collaborations?

Thank you very much for the compliment. I think that the urge to collaborate is part of my personality. Some people prefer to stay in an area and do deep work there, and other people enjoy collaborations. I was extremely lucky to have wonderful colleagues from industry, statistics, and machine learning; the collaborations just developed by themselves and I didn't have to do anything special to nurture them.

How do you learn the basics of so many techniques in so many different fields, and bring them to bear on database problems?

The criterion is that if I see a method being applied two or three times, or being reinvented two or three times, then it is probably a method that could have application also in databases. That is what happened with fractals. Now we are trying to do the same with singular value decomposition, with independent component analysis, because these techniques have the potential for deep influence in many disciplines.

I have also been told that you are "the nicest guy in the whole wide world" and "completely altruistic," so I am sure that that also has something to do with the success of your collaborations. Someone suggested that I ask you how you always manage to smile!

You have two brothers who are also computer scientists, and with them you wrote what has become known as the "Faloutsos cubed paper," a very influential paper about power laws that hold with respect to the internet topology of 1997-8. What are these power laws? Do they still hold today? Why should we in the database community care about them?

The power laws still hold today. I have graphs that show the power laws holding for several years after 1997-8. And they seem to hold not only for computer networks! It is like Zipf's Law: some words appear very often, and most vocabulary words appear only once or never at all. The same is true for internet connections: there are a few nodes that are very popular. Everybody wants to connect to AT&T or IBM or Sprint, and nobody wants to connect to a tiny little ISP. The same is true also for company sizes. There are huge companies with a quarter million employees, but the vast majority of companies have only one or two employees. So these power laws will hold for not only for networks, as in the Faloutsos cubed paper, but also for many other settings---and for several centuries, not only recently.

Where should we apply the power laws in the database field?

We can apply them for selectivity estimation, for histograms. Yannis Ioannidis got the VLDB 2003 Best Paper Award for his paper on histograms. Histograms are superbly successful exactly because of these Zipf distributions: if you keep the frequency counts of the few most important attributes, then the rest don't matter that much.

Power laws have a very deep connection with fractals. Power laws, fractals, and self-similarity appear in many settings, and we can use self-similarity to battle the dimensionality curse. In databases and in data mining, if we have a lot of attributes, we say we have the problem of the dimensionality curse. High dimensionality is a problem because the running times of most of the data mining algorithms explode exponentially if there are many attributes. But it's not the dimensionality that matters; it's the fractal, the intrinsic dimensionality, of the data set that matters. The fractal dimensionality is usually much lower, exactly because of the skewed distribution of the importance of attributes. There may be a hundred attributes, but only the first three or four or five of them are the most important ones, and therefore the problem is not as hard as we would fear.

*So, **fractal** is the answer. What is the new and most important question?*

There are a lot of questions that will benefit from fractals, such as analysis of graphs and social networks. There are definitely power laws on almost any type of graph we get, and probably self-similarity also. Social networks (who knows whom), biological networks, and food web networks (who eats whom) all have self-similarity and fractals. The sensor time series analysis that we are working on also has self-similarity. Time series have burstiness, which can be very well described with self-similarity. You have silent periods, explosions, bigger silences, bigger explosions, as opposed to the standard Poisson distribution that says that every now and then you have an event happening. No, instead events are very clustered and self-similar. So I think that fractals will be the answer to multiple questions, not only one.

The field of data mining is young, vibrant, and quickly evolving. What do you see as the major future directions in data mining---where is the field going?

That's another very good question. Definitely there are a lot of possibilities with the web, computer networks, social networks, biological networks, regulatory networks; there is a lot of emphasis on networks. Time series analysis also, because we'll be flooded with measurements from sensors, and we want to find patterns there. We want to find intrusions if we have a network, and so we measure how many packets or how many pings we get per time unit. Bioinformatics should also become a hot area. Actually, it *is* a hot area.

It sounds like you are saying that the field will be concentrating on the application areas for a while.

Yes, but that is a biased opinion, because I'm more oriented toward the practical areas. I'm sure my more theoretically-minded and statistics-oriented colleagues from data mining will have a different opinion. They will be looking forward to studying deeper mathematical problems. So mine is a biased opinion from an application person.

With social networks, what kind of patterns are we mining? What are we looking for?

We want to find common patterns, like the way Jiawei Han is trying to do for AIDS virus molecules, where he is trying to see what submolecules occur often. We want to find what groups of people occur often in a company network and figure out whether the appearance of a particular group is destructive or constructive for the department. We want to figure out which are the outlier edges. So if we have, say, a group of researchers who usually don't talk to each other, then if we see edges between them, these are important edges. These edges are either suspicious, because they shouldn't be happening, or else they are very valuable because these are the bridges that make the department work harmoniously together. The problem with data mining is that we are not looking for something specific. We try to look for something that we don't know yet, a pattern that will help us compress this data set.

The data mining field includes people with backgrounds in statistics, AI, and databases. I have heard that people from one area can't understand the conference talks of people from the other areas. I have heard that a statistics person said to you, after one of your talks, that he could always come up with a query that would break your index structure, so what was the point of having the index? What will happen to the field, with this uneasy alliance of subdisciplines that don't really understand each other?

I think what will happen is what is happening already: conferences try to put all these people in the same room and after the first few uneasy years, they will understand each other's mentality, as is the case now. So in database classes, we are teaching about Chi-squared tests because it's useful for statistics, and I'm sure that statisticians are teaching about B-tree indices. I don't remember the specifics of the situation that you mentioned, but there is a lot of cross-fertilization. Yes, the first few years will be uneasy, but eventually the goal is worth the initial pain.

It seems that many people from Greece do database research. What are your views on this heritage---is it an opportunity, a burden, or something else altogether?

I think that it's a happy coincidence. It's the 80/20 law and fractals in action, a clustering effect. A few database professors came back to Greece when I was an undergraduate, like Dennis Tsichritzis, and of course they were all enthusiastic about databases. Then we all went out to the states or Canada, and the same thing repeated itself a few times, creating exponential growth. Now we have a huge number of Greek students and Greek professors doing database research. I'm sure this is the case with other nations too; there are a lot of Indian and Israeli database professors too. So it's a happy coincidence.

You first came to Carnegie Mellon as a sabbatical visitor, and then stayed on to become a professor there. What was it like to make the transition from Maryland, a database-oriented department, to CMU, which had never had a database faculty member when you arrived? How do you cope with having to justify your entire discipline to everyone you meet?

Actually, it was a very pleasant transition, because Carnegie Mellon was actively trying to build a database group. So yes, I had to do some justification, but it was mainly what I had to do was education. I had to tell people that a database is not a collection of information, it's a collection of tables with SQL on top. People

at Carnegie Mellon are extremely nice and very cross-disciplinary; they are all hand picked to be cross-disciplinary by the hiring process. So it was very easy.

How did you convince them that databases were important? Were you arguing that databases are of economic importance, or just intellectually interesting?

I didn't have to argue at all, because they were already convinced that database research is important. That's why they invited me.

Natassa Ailamaki says she has had to justify her discipline over and over again.

We have had more to *explain* than to justify, because they all have large data sets and they want to do something with them. Even during my year as a visitor, people would say, "Oh, so you know databases, great! I have this problem, can you help me? I have time series for monkey brains. How can I store them and try to find similarities?" They want to do neurobiology and figure out how the monkey brains operate when you show them visual stimuli. So it was not a matter of justification, it was a matter of a quick crash course.

I have heard that you take many sabbaticals, and that you like to spend them in a particular way. What recommendations do you have for faculty members considering a sabbatical?

I think sabbaticals at an industrial lab are very valuable because they help us get in touch with real problems, real customers, and get in touch with the trenches. That's why I spent two sabbaticals at IBM with Rakesh Agrawal and Bill Cody, and at AT&T with Avi Silberschatz and H. V. Jagadish when they were both at AT&T. So my suggestion is to try to find out what real customers are complaining about.

To really be in touch with the real customers, wouldn't you need to go a development group?

Actually, no, because the collaborators have direct contact with customers or with other people within the company who are in direct contact with customers. It's close to customers but not extremely close, not face-to-face interviews with customers, but their complaints percolate eventually to the research labs.

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

I think the major advice is for people to make sure that they enjoy what they are doing. If you find a topic interesting, then other people will find it interesting too. Before tenure, of course it's important to focus on the rules of the game: if the university wants journal publications, let's make sure that we have the appropriate number, and so on.

After tenure, I think people are free to do what they enjoy most, which is a matter of taste. Personally, I prefer to work on problems that have practical importance and also can use some nice theoretical solutions. Other people prefer to focus on practical issues; as long as the problem is important for companies or society, then they work on it. And at the other extreme, some people work on completely theoretical issues that may or may not have applications. I think all three modes are valuable. People should pursue whatever keeps them up at night.

The problem that I see with your philosophy is that after you get tenure and have the freedom to do whatever you want, you still have those grad students. In order to get a job, many of them feel that they have to behave a lot like an assistant professor in producing all these papers. So how do you get off the treadmill once you get tenure?

I don't think you get off the treadmill ever! Unfortunately, people think, “If I get tenure I’ll get relaxed.” No, nobody will relax. Just more freedom and more peace of mind, but still it’s still the same amount of work. I think the amount of work the person is doing when he or she is a graduate student will be the same until this person retires. It’s a matter of state, it’s a matter of mentality.

We shouldn’t tell them that, should we? Won’t all the grad students who read this get depressed?

I don’t think so. It is true, they are smart and they see their professors (assistant, associate, full, emeritus) working 10 and 12 and more hours a day. But this is *fun* work, this is something that people enjoy doing, and I don’t think it’s bad.

That’s a good lead-in to my next question: if you magically had enough extra time at work to do one thing that you are not doing now, what would that thing be?

Nothing different.

More of the same?

More of the same: playing with drafts, collecting data sets, trying to find patterns, trying to figure out what is the next best tool to use for all the problems mentioned before.

Among all your past research, what is your favorite piece of work?

It is probably the PODS 94 paper on how to use fractals to characterize non-uniformity of a cloud of points, so that we can figure out the performance of R-trees and other spatial access methods.

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

Maybe get better organized. For now, things are not so well organized.

Sometimes people have their secretary or their postdoc do all their organizing, to keep them on track. Have you tried that?

No, I should. That’s a good idea.

I have been told that you are a resource for jokes for the Greek database community. Can you tell us a joke to conclude our interview?

Of course! The shortest joke I know: I'm an atheist, thank God!

Thank you very much.

Thank you very much!

Reminiscences on Influential Papers

Kenneth A. Ross, editor

Unfortunately, this will be my last influential papers column. I've been editor for about five years now (how time flies!) and have enjoyed it immensely. I've always found it rewarding to step back and look at why we do the research we do, and this column makes a big contribution to the process of self-examination. Further, I feel that there's a strong need for ways to publicly and explicitly highlight "quality" in papers. Criticism is easy, and is the more common experience given the amount of reviewing (and being reviewed) we typically engage in. I look forward to seeing this column in future issues.

Ken Ross.

AnHai Doan, University of Illinois, anhai@cs.uiuc.edu.

[Richard Hamming: You and Your Research. Seminar Talk at Bell Communication Research, 1986; transcribed by J. F. Kaiser.]

In this paper Richard Hamming discusses what it takes to do great research. He considers multiple topics, ranging from well-known ones such as problem selection, courage, hard work, and communication skills, to less familiar ones such as the need to tolerate ambiguity. The topics were discussed in an engaging manner, and vividly illustrated with personal anecdotes. The paper as a whole was a lot of fun to read. It can be found on the Internet (and a 3-page summary is available by googling for "striving for greatness Hamming").

I first read this paper during my Ph.D. years, and have periodically reread it ever since. The paper has influenced me in three ways. First, it confirmed some of my vague ideas about the research process, and suggested new "tricks-of-the-trades". Second, it has and continues to inspire me to do my best in research. Hamming stresses the need to continually ask ourselves: "What am I doing? And what are the important problems in my field?". Reading this part periodically does help me to keep an eye on the big picture, and to put day-to-day concerns, such as the strong pressure to publish, in perspective.

Finally, the "imperfections" of this paper do provoke me to think deeper about the research process. For example, Hamming claims that first-class research is worth the effort because it is as good as wine, the opposite sex, and song put together. I found this claim ...um... not terribly convincing, but it did make me think about what makes many of us "suffer" long hours in this business. My own theory (developed after some wine) is that research is an *art*, and researchers are artists. Artists do not have jobs, but rather *callings*: the call to create lasting, beautiful messages, and to communicate them to (influence) others. Such messages are packaged in various forms: novels for writers, paintings for painters, and papers (and students) for us. Hence, as artists, we endure as we strive to create lasting work.

Sihem Amer-Yahia, AT&T Laboratories, sihem@research.att.com.

[Janet L. Wiener, Jeffrey F. Naughton: OODB Bulk Loading Revisited: The Partitioned-List Approach. VLDB 1995: 30–41,]

I read this paper when I was at INRIA, right at the beginning of my Ph.D. I did not know then what it meant to do database research. All I knew was the relational model, SQL and the existence of database systems. This paper had a great impact on my research in different ways.

First, although the focus its was on loading data into object-oriented databases, the paper connected what I then knew of databases altogether. I understood that data queried using SQL was not sitting in a database

by miracle as it needed to be loaded from the outside world. I also understood that something beyond SQL was needed to do it.

Second, the paper connected the database industry world with the world of database research, which at that time, gave me a sense for research and, meant that designing loading algorithms was a tough research question.

Third, the algorithms proposed in the paper were not only implemented and tested inside Shore, a real system, but several times in the paper, the authors compared their design and performance with other loading solutions in other real systems.

Finally, although I had not fully realized it then, the paper and its preceding version, published in VLDB 1994, are what I consider thorough and really informative database performance papers. It looks at a real, apparently simple problem that many of us thought is solved and that many of us still encounter, maybe without realizing it. The algorithms are very elegant and make use of all the features of a database system such as sorting and indexing. The experimental evaluation is full of exciting details and looks at every aspect of loading: CPU time, buffer pool space, I/Os. Finally, the authors did a not-so-typical thing which is to learn from the results of their experiments and propose in the same paper, new ideas and actually give enough details to implement them as opposed to throwing in some future work directions. I really appreciate that now as it became the basis for my Ph.D work.

This paper convinced me to look at algorithmic issues in database systems. Unfortunately, I seem to not have managed to learn how to do such a detailed database performance evaluation yet!

Changes to the *TODS* Editorial Board

Richard Snodgrass

rts@cs.arizona.edu

The December issue should be out soon, if it isn't already. It is a special issue of papers invited from the SIGMOD'04 and PODS'04 conferences. I thank the authors, associate editors, and reviewers (some from the SIGMOD and PODS program committees) for writing, reviewing, revising, reviewing again, and then getting into final form these papers in the amazingly short time of eighteen months. This is a very nice issue, with seven substantive papers.

The 2005 volume (volume 30) of *TODS* is the largest one yet, by far, and the first one over 1000 pages (over 1100 pages, actually). Included in this volume, in the September 2005 issue, is one of Alberto Mendelzon's last papers, "Capturing summarizability with integrity constraints in OLAP." Alberto was a *TODS* Associate Editor for four years, and before that worked with me as SIGMOD Information Director. I greatly miss him and his gentle smile.

Submissions to *TODS* have doubled over the past three years. I'm very pleased to announce the appointment of four new Associate Editors, bringing the Editorial Board to 23.[1]

Phokion Kolaitis works on logic in databases. Current research interests include data integration and interoperability, metadata management, query languages, and deductive databases.

Ken Ross's current interests include query processing and optimization, particularly in the context of database systems running on new and/or nontraditional hardware. He also has projects in several application domains, including archeology, architecture, and biology.

Pierangela Samarati's research interests are in the areas of access control policies, models, languages, and systems; data security and privacy; information system security; and information protection in general.

S. Sudarshan's main research interests include query optimization, keyword queries on structured and semi-structured data, and fine-grained access control. He is also working on tools for supporting database application development.

We're very fortunate that such accomplished scholars are willing to invest their valuable time to handle manuscripts.

[1] http://www.acm.org/tods/Editors.html