SIGMOD RECORD

SIGMOD Record – December 2006 – Volume 35 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

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, Michael Carey, and Jeffrey D. Ullman.

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, Hongjun Lu, and Tamer Özsu.

Why is Schema Matching Tough and What Can We Do About It?

Avigdor Gal
Technion – Israel Institute of Technology
avigal@ie.technion.ac.il

ABSTRACT

In this paper we analyze the problem of schema matching, explain why it is such a "tough" problem and suggest directions for handling it effectively. In particular, we present the monotonicity principle and see how it leads to the use of top-K mappings rather than a single mapping.

1. INTRODUCTION

Schema matching is the task of matching between concepts describing the meaning of data in various heterogeneous, distributed data sources (*e.g.* database schemata, XML DTDs, HTML form tags, *etc.*). Schema matching is recognized to be one of the basic operations required by the process of data integration [4], and thus has a great impact on its outcome. Schema mappings (the outcome of the matching process, as will be defined in Section 2) can serve in tasks of generating global schemata, query rewriting over heterogeneous sources, duplicate data elimination, and automatic streamlining of workflow activities that involve heterogeneous data sources. As such, schema matching has impact on numerous applications. It impacts business, where company data sources continuously realign due to changing markets. It also impacts life sciences, where scientific workflows cross system boundaries more often than not.

Despite two decades of research in this area, summarized in surveys (*e.g.*, [21, 7, 23]) and various online lists (*e.g.*, OntologyMatching[1], Ziegler[2], DigiCULT[3], SWgr[4]) schema matching still seems to involve ad-hoc solutions with only a few works that involve foundational principles of schema matching [4, 17, 15, 11, 2].

In 2001, Maurizio Lenzerini claimed that "Data Integration Is Harder Than You Thought" [14]. Here we shall analyze the problem of schema matching, which is a subproblem of data integration, explain why it is such a "tough" problem and suggest directions for handling it effectively.

2. SCHEMA MATCHING BASICS

Due to its cognitive complexity [6], traditionally schema matching has been performed by human experts [13]. As the process of data integration has become more automated, the ambiguity inherent in concept interpretation has become one of the main obstacles to effective schema matching. For obvious reasons, manual concept reconciliation in dynamic environments (with or without computer-aided tools) is inefficient and at times close to impossible. Introduction of the Semantic Web vision [3] and shifts toward machine-understandable Web resources and Web services have made even clearer the vital need for automatic schema matching.

Although these tools comprise a significant step towards fulfilling the vision of automated schema matching, it has become obvious that the user must accept a degree of imperfection in this process [11]. A prime reason for this is the enormous ambiguity and heterogeneity of data description concepts: It is unrealistic to expect a single mapping engine to identify the correct mapping for any possible concept in a set. Another (and probably no less crucial) reason is that "the syntactic representation of schemas and data do not completely convey the semantics of different databases" [19]; *i.e.*, the description of a concept in a schema can be semantically misleading. Therefore, managing uncertainty in schema matching has been recognized as the next issue on the research agenda in the realm of data integration [15].

As a basis for our discussion we next layout a generic model for schema matching that serves the needs of this paper. However, the reader should not consider this model to be complete or unique. Let S_1 and S_2 be two schemata, defined using some data model (*e.g.*, relational or ontological), with n_1 and n_2 attributes, respectively. We set no particular constraints on the nature of attributes. Therefore, attributes can be as simple as relational schema attributes, or complex, *e.g.*, sub-trees in an XML schema. A common representation of the schema matching problem in the literature uses a bipartite graph $G = (V, E)$, where nodes represent attributes, each side of the graph represents a different schema (*i.e.*, $V = S_1 \cup S_2$), and an edge (v, u) represents a possible mapping between attributes. A schema mapping in this setting is a subset $E' \subseteq E$.

3. MODELING UNCERTAINTY IN SCHEMA MATCHING

Melnik and Bernstein [4, 17] have proposed the *Match* abstraction as a basic tool for model management. Match operates on schemata and returns a mapping $E' \subseteq E$.

To represent the uncertainty inherent in the matching process, one can extend the bipartite graph to use labeled edges, where a label of an edge can have the semantics of a "level of certainty." Therefore, a label can be a function $\omega : E \rightarrow [0, 1]$ where a label of 1 between two nodes represents the highest level of certainty regarding the attribute mapping.

It is common to compute the uncertainty of a schema

[1]http://www.ontologymatching.org/
[2]http://www.ifi.unizh.ch/~pziegler/IntegrationProjects.html
[3]http://www.digicult.info/pages/resources.php?t=10
[4]http://www.semanticweb.gr/modules.php?name=News&file=categories&op=newindex&catid=17

matching from the level of uncertainty of its components. Therefore, given a mapping (E', one can define a schema mapping level of certainty as a function $\Omega : 2^E \rightarrow [0,1]$. Examples of Ω include weighted average, dice [8], and others (*e.g.*, by comparing top-2 mappings [5]).

So far, no limitations were set on the set of edges in E'. A typical cardinality constraint of $1:1$ sets a constrains as follows: $\forall v \in V, (v,u) \in E' \rightarrow \forall w \in V \backslash \{u\}, (v,w) \nrightarrow E'$. It is worth noting that only limited research is currently devoted to other types of cardinality constraints, such as $1:n$, $n:1$, and $n:m$ (*e.g.*, [24]). The interested reader is referred to [9] for a discussion and analysis of this phenomenon.

While modeling uncertainty in schema matching provides a nice formal framework, many questions remain unresolved. How can one choose a good heuristic for determining the labeling of edges (ω)? Are there "good" and "bad" Ω functions? The next section presents the monotonicity principle as a mechanism for providing insights into the selection process of the ω and Ω functions.

4. THE MONOTONICITY PRINCIPLE

The evaluation of schema mappings (the outcome of schema matching) is typically performed with respect to some "golden rule" mapping, as given (at least conceptually) by a domain expert. We denote such a mapping to be the *exact mapping*. Clearly, such expert opinions are not readily available when matching schemata (otherwise, one can simply use the expert opinion). Therefore, empirical evaluation is typically performed on a limited set of schemata to "get the feeling" on the performance of a matching algorithm. Two metrics (and their combinations), borrowed from the area of Information Retrieval, namely *precision* and *recall*, were used for the empirical evaluation of performance. Assume that out of the $n_1 \times n_2$ attribute mappings, there are $c \leq n_1 \times n_2$ correct attribute mappings, with respect to the expert mapping. Also, let $t \leq c$ be the number of mappings, out of the correct mappings, that were chosen by the matching algorithm and $f \leq n_1 \times n_2 - c$ be the number of incorrect such attribute mappings. Then, precision is computed to be $\frac{t}{t+f}$ and recall is computed as $\frac{t}{c}$. Clearly, higher values of both precision and recall are desired. From now on, we shall focus on the precision measure, denoting by $p(E')$ the precision of a schema mapping E'. Extensions that include the recall measure as well are left open for future research.

We observe that precision takes its values from a discrete domain in $[0,1]$. Therefore, one can create equivalence schema mapping classes on 2^E, the power set of G's edges. Two mappings E' and E'' belong to a class p if $p(E') = p(E'') = p$, where $p \in [0,1]$. Let us consider now two mappings, E' and E'', such that $p(E') < p(E'')$. For each of these two mappings we can compute their schema mapping level of certainty, $\Omega(E')$ and $\Omega(E'')$. We say that a matching algorithm is *monotonic* if for any two such mappings $p(E') < p(E'') \rightarrow \Omega(E') < \Omega(E'')$.

Clearly, a monotonic matching algorithm can easily identify the exact mapping. Let E^* be the exact mapping, then $p(E^*) = 1$. For any other mapping E', $p(E') \leq p(E^*)$, since p takes its values in $[0,1]$. Therefore, if $p(E') < p(E^*)$ then from monotonicity $\Omega(E') < \Omega(E^*)$. All one has to do then is to devise a method for finding a mapping E^* that maximizes Ω.[5] In fact, this is one of the two most common methods for identifying the exact mapping nowadays [8, 11, 5]. The other common method, adopted in [18, 12] and others, is to only determine the values of ω automatically, allowing the user to identify the exact (schema) mapping from the individual attribute mappings.

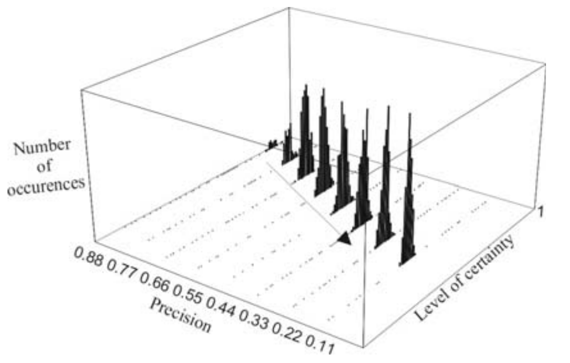


Figure 1: Illustration of the monotonicity principle

Figure 1 provides an illustration of the monotonicity principle using a mapping of "Absolute Agency" with "Adult Singles," both taken from the dating and matchmaking domain, using the combined algorithm, as provided by OntoBuilder.[6] Given a set of mappings, each value on the x-axis represents a class of schema mappings with a different precision. The z-axis represents the level of certainty. Finally, the y-axis stands for the number of schema mappings from a given precision class and with a given level of certainty.

Two main insights can be derived from Figure 1. First, the level of certainty of mappings within each schema mapping class form a "bell" shape, centered around a specific level of certainty. Such a behavior indicates a certain level of robustness of a schema matcher, assigning similar certainty levels to mappings within each class. Second, the "tails" of the bell shapes of different classes overlap. Therefore, a schema mapping from a class of a lower precision may receive a higher level of certainty than a mapping from a class of a higher precision. This, of course, contradicts the monotonicity definition. In fact, our experience with various schema matching algorithms and various real-world schemata shows that no matcher we have encountered is shown (even empirically) to be monotonic. However, the first observation serves as a motivation for a definition of a statistical monotonicity, first introduced in [11]:

DEFINITION 1 (STATISTICAL MONOTONICITY). *Let $\mathcal{E} = \{E_1, E_2, ..., E_m\}$ be a set of mappings over schemata S_1 and S_2 with n_1 and n_2 attributes, respectively, and define $n = \max(n_1, n_2)$. Let $\mathcal{E}_1, \mathcal{E}_2, ..., \mathcal{E}_{n+1}$ be subsets of $\mathcal{E}$ such that for all $1 \leq i \leq n+1$, $E \in \mathcal{E}_i$ iff $\frac{i-1}{n} \leq p(E) < \frac{i}{n}$. We define M_i to be a random variable, representing the level of certainty of a randomly chosen mapping from $\mathcal{E}_i$. $\mathcal{E}$ is* statistically monotonic *if the following inequality holds for any $1 \leq i < j \leq n+1$:*

$$\bar{\Omega}(M_i) < \bar{\Omega}(M_j) \qquad (1)$$

[5]In [11], where the monotonicity principle was originally introduced, it was shown that while such a method works well for fuzzy aggregators (*e.g.*, weighted average) it does not work for t-norms such as min.

[6]http://ie.technion.ac.il/OntoBuilder

where $\bar{\Omega}(M)$ stands for the expected value of M.

Intuitively, a schema matching algorithm is statistically monotonic with respect to given two schemata if the expected certainty level increases with precision. Statistical monotonicity can assist us in explaining certain phenomena in schema matching and also to serve as a guideline in finding better ways to use schema matching algorithms.

5. FROM STATISTICAL MONOTONICITY TO TOP-K SCHEMA MAPPINGS

Consider the set of all possible mappings between two schemata, ranked in a decreasing order of $\Omega(\cdot)$ of some (statistically) monotonic matcher. Assume that this set of mappings is statistically monotonic. Looking at the top-ranked mapping (the best mapping E'), is it the exact mapping E^*? Not necessarily. Although $\bar{\Omega}(M_i) < \bar{\Omega}(M_n)$ for all $1 \leq i < n$, for specific instances $\Omega(E') > \Omega(E^*)$ is a valid option, given the statistical nature of our definition. Therefore, an immediate observation from this analysis is that if one limits oneself to a matching task that identifies the best mapping, one may not identify the exact mapping. This is a well-known fact in the schema matching research area, since rarely one has a precision and recall of 1, indicating that $E' = E^*$. Therefore, this model serves as a justification, in a retrospect to a tough reality (recall that the first part of the title of this paper is "Why is Schema Matching Tough").

An additional observation goes as follows. Given the monotonic nature of our matcher, it seems likely that the number of mappings E' satisfying $\Omega(E') > \Omega(E^*)$ is small with respect to the total number of possible mappings. To motivate this observation, consider once more Figure 1 and note that a mapping E' for which $\Omega(E') > \Omega(E^*)$ must come from the "upper tail" of the mapping groups with lower precision. The chance that such a mapping will indeed receive a higher similarity measure decreases with group precision. Therefore, the exact mapping is likely to be found in the top-K mappings, where K depends on the distribution of similarity values of the precision groups, but is likely to be much smaller than the set of all possible mapping.

Rank	Combined algorithm
0	71%
1-10	19%
11-99	10%
>100	0%
Average rank	7

Table 1: Exact mapping positioning with respect to the best mapping

Table 1 presents an empirical analysis, taken from [11], summarizing the positioning of the exact mapping using a statistically monotonic matcher. A rank of 0 means that the algorithm was successful in identifying the exact mapping as the best mapping. Other ranks show the positioning within all possible mappings ($9! = 362,880$). Even if this matcher fails to identify the exact mapping as the best mapping, it was still ranked high, saving one the need to possibly iterate over all permutations.

Following this observation, a matching process can be devised iteratively, where in each iteration a schema mapping is tested for correctness (*e.g.*, by sending a form to a server). In case of a failure, the next best mapping is being constructed. An algorithm for generating top-K mappings was presented in [1], based on Murty's assignment algorithm [20].

In [10], another use of top-K mappings was suggested. Previously, we have assumed that a matcher can find the exact mapping. Such an assumption may not be valid. For example, assume that our matching algorithm solves a maximum weight bipartite graph problem to identify mappings. Such an algorithm aims at adding as many attributes as possible to a mapping. However, in many real-world cases, some attributes cannot be mapped. Such a scenario is typically handles by setting a threshold so that attribute mappings with low certainty level cannot be included in a schema mapping. Alas, thresholds are not easily tuned [22]. Therefore, the algorithm proposed in [10] suggests to generate a dynamic threshold by analyzing **simultaneously** a set of top-K mappings. According to our observations, monotonic matchers tend to rank high schema mappings with many correct attribute mappings. Therefore, the algorithm keeps attribute mappings only if they appear a sufficient number of times (a threshold t) in the top-K mappings.

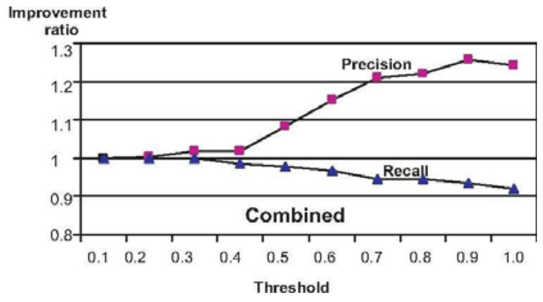


Figure 2: Precision and Recall for Stability analysis with $K = 10$

Figure 2 presents the average change to precision and recall for different thresholds over 43 real data pairs, as presented in [10]. K was set to 10. For this data set, precision increases (in general) up to $t = 0.9$ with the increased threshold. Recall demonstrates a monotonic decrease with the increased threshold. Such a phenomenon accords with our initial intuition and is expected for monotonic matchers. A closer look at the amount of improvements reveals that the matcher provides an increase of 25.6% (with $t = 0.9$). As for recall, it decreases by a maximum of 8%.

6. DISCUSSION AND CONCLUSIONS

In this paper we have introduced the principle of monotonicity as a theoretical principal in schema matching and showed the relationships between matcher monotonicity and its ability to generate useful mappings. A few attempts at setting theoretical foundation for schema matching exist in the literature. The seminal work of Melnik and Bernstein [4, 17] has been discussed already in Section 3.

A recent work on representing and reasoning about mappings between domain models was presented in [15]. This work provides a model representation and inference analysis. Managing uncertainty was recognized as the next step on the research agenda in this area and was left open for a future research. Our work fills this gap in providing a model that represents the uncertainty (as an imprecision measure) in the matching process outcome.

Soundness of schema matching methods was discussed in [2]. There, matching correctness was defined using *pragmatic competence*, the ability to make decisions that are sound with respect to the semantics of the problem. The monotonicity principle can be viewed as a method for refining pragmatic competence using quantified methods.

Studying the uncertainty inherent to the schema matching process (which is the "tough" part of schema matching) is an ongoing research task (see, *e.g.*, [16]). More matchers are needed, utilizing top-K mappings. Also, a more refined classification of monotonic matchers is needed, based on aspects such as application domain and the amount of variance of certainty level values within each precision group.

Acknowledgments

Many thanks to Phil Bernstein, George Fletcher, and Amit Sheth for useful comments and discussions.

7. REFERENCES

[1] A. Anaby-Tavor. Enhancing the formal similarity based matching model. Master's thesis, Technion-Israel Institute of Technology, May 2003.

[2] M. Benerecetti, P. Bouquet, and S. Zanobini. Soundness of schema matching methods. In *Proceedings of ESWC 2005*, pages 211–225, 2005.

[3] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic Web. *Scientific American*, May 2001.

[4] P.A. Bernstein and S. Melnik. Meta data management. In *Proceedings of the IEEE CS International Conference on Data Engineering*. IEEE Computer Society, 2004.

[5] A. Bilke and F. Naumann. Schema matching using duplicates. In *Proceedings of the IEEE CS International Conference on Data Engineering*, pages 69–80, 2005.

[6] B. Convent. Unsolvable problems related to the view integration approach. In *Proceedings of the International Conference on Database Theory (ICDT)*, Rome, Italy, September 1986. In *Computer Science*, Vol. 243, G. Goos and J. Hartmanis, Eds. Springer-Verlag, New York, pp. 141-156.

[7] H. Do, S. Melnik, and E. Rahm. Comparison of schema matching evaluations. In *Proceedings of the 2nd Int. Workshop on Web Databases (German Informatics Society), 2002.*, 2002.

[8] H.H. Do and E. Rahm. COMA - a system for flexible combination of schema matching approaches. In *Proceedings of the International conference on very Large Data Bases (VLDB)*, pages 610–621, 2002.

[9] A. Gal. On the cardinality of schema matching. In *IFIP WG 2.12 and WG 12.4 International Workshop on Web Semantics (SWWS)*, pages 947–956, 2005.

[10] A. Gal. Managing uncertainty in schema matching with top-k schema mappings. *Journal of Data Semantics*, 6:90–114, 2006.

[11] A. Gal, A. Anaby-Tavor, A. Trombetta, and D. Montesi. A framework for modeling and evaluating automatic semantic reconciliation. *VLDB Journal*, 14(1):50–67, 2005.

[12] B. He and K.C.-C. Chang. Making holistic schema matching robust: an ensemble approach. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*, pages 429–438, 2005.

[13] R. Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 51–61. ACM Press, 1997.

[14] M. Lenzerini. Data integration is harder than you thought. In C. Batini, F. Giunchiglia, P. Giorgini, and M. Mecella, editors, *Cooperative Information Systems, 9th International Conference, CoopIS 2001, Trento, Italy, September 5-7, 2001, Proceedings*, volume 2172 of *Lecture Notes in Computer Science*, pages 22–26. Springer, 2001.

[15] J. Madhavan, P.A. Bernstein, P. Domingos, and A.Y. Halevy. Representing and reasoning about mappings between domain models. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI)*, pages 80–86, 2002.

[16] M. Magnani, N. Rizopoulos, P. McBrien, and D. Montesi. Schema integration based on uncertain semantic mappings. In *ER*, pages 31–46, 2005.

[17] S. Melnik. *Generic Model Management: Concepts and Algorithms*. Springer-Verlag, 2004.

[18] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings of the IEEE CS International Conference on Data Engineering*, pages 117–140, 2002.

[19] R.J. Miller, L.M. Haas, and M.A. Hernández. Schema mapping as query discovery. In A. El Abbadi, M.L. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, and K.-Y. Whang, editors, *Proceedings of the International conference on very Large Data Bases (VLDB)*, pages 77–88. Morgan Kaufmann, 2000.

[20] K.G. Murty. An algorithm for ranking all the assignments in order of increasing cost. *Operations Research*, 16:682–687, 1968.

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

[22] M. Sayyadian, Y. Lee, A. Doan, and A. Rosenthal. Tuning schema matching software using synthetic scenarios. In *Proceedings of the International conference on very Large Data Bases (VLDB)*, pages 994–1005, 2005.

[23] P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. *Journal of Data Semantics*, 4:146 – 171, December 2005.

[24] W. Su, J. Wang, and F. Lochovsky. Aholistic schema matching for web query interfaces. In *Advances in Database Technology - EDBT 2006, 10th International Conference on Extending Database Technology, Munich, Germany, March 26-31, 2006, Proceedings*, pages 77–94, 2006.

TPCC-UVa: An Open-Source TPC-C Implementation for Global Performance Measurement of Computer Systems

Diego R. Llanos
Departamento de Informática
Universidad de Valladolid, Spain.
diego@infor.uva.es.

Abstract

This paper presents TPCC-UVa, an open-source implementation of the TPC-C benchmark version 5 intended to be used to measure performance of computer systems. TPCC-UVa is written entirely in C language and it uses the PostgreSQL database engine. This implementation includes all the functionalities described by the TPC-C standard specification for the measurement of both uni- and multiprocessor systems performance. The major characteristics of the TPC-C specification are discussed, together with a description of the TPCC-UVa implementation, architecture, and performance metrics obtained. As working examples, TPCC-UVa is used in this paper to measure performance of different file systems under Linux, and to compare the relative performance of multi-core CPU technologies and their single-core counterparts.

Keywords: On-line transaction processing, TPC, performance measurement.

1 Introduction

Workload characterization in order to measure system performance is a major topic in the field of Computer Architecture. Many different benchmarks have been proposed to simulate real working conditions of both existing and proposed systems. Those benchmarks can be classified in terms of their corresponding application domains and their execution characteristics.

The most popular benchmarks are related with numerical processing, such as the SPEC CPU2000 benchmark suite [4], the NAS Parallel Benchmark [7] and the OLDEN benchmarks [10], among others. These benchmarks include many common characteristics of real scientific workloads, and some of them can be executed in both sequential and parallel computing environments. These benchmarks are designed to challenge the CPU and memory subsystem capabilities of the systems under test. However, they do not take into account other aspects of the system architecture, such as process management or I/O subsystem.

Database benchmarks, on the other hand, allow to study not only CPU and memory hierarchy performance, but the global performance of a system. These benchmarks use a synthetic workload against a database engine, measuring the performance of the system in terms of the number of transactions completed in a given period of time. One of the main advantages of this class of benchmarks is that results are very relevant to financial, commercial and corporative fields, where this type of applications is dominant.

The TPC-C benchmark, designed by the Transaction Processing Performance Council [1], simulates the execution of a set of both interactive and deferred transactions. This workload is representative of an OLTP (On-line Transaction Processing) environment, with features such as transaction queries and rollback. These capabilities makes the TPC-C benchmark specification a de-facto standard for measuring server performance. Most vendors publish performance values for their systems, allowing the consumer to accurately compare different architectures.

The Transaction Processing Performance Council only distributes a requirements specification for the TPC-C benchmark. Following this specification, vendors may implement and run a TPC-C benchmark, needing the approval of the TPC consortium to publish its performance results [3]. Unfortunately, there is not an official TPC-C benchmark implementation available for research purposes.

In this paper we describe TPCC-UVa [5], an unofficial, open-source implementation of the TPC-C benchmark version 5. The purpose of TPCC-UVa is to be used as a research benchmark for the scientific community. The TPCC-UVa benchmark is written entirely in C language, and it uses the PostgreSQL database engine. This implementation has been extensively tested on Linux systems, and it is easily portable to other platforms. TPCC-UVa source code is freely distributed from the project website[1]. This makes easy to use it for the performance measurement and behavior of real systems or in the context

[1] http://www.infor.uva.es/~diego/tpcc-uva.html.

of a simulation environment such as Simics [6]. As an example, TPCC-UVa has been recently used in the performance measurement of both existent and experimental file systems [8].

The TPCC-UVa implementation includes all the characteristics described in the TPC-C standard specification, except support for price/performance comparison. The reason is that TPCC-UVa is only intended to be used for measuring performance in research environments. It is important to highlight the fact that TPCC-UVa is not an implementation approved by TPC, and the results of the execution of TPCC-UVa, in particular its performance parameter (tpmC-uva), should not be compared with the performance values obtained by official implementations of TPC-C.

The rest of the article is organized as follows. Section 2 describes the main characteristics of the TPC-C benchmark specification. Section 3 presents the TPCC-UVa implementation, describing its architecture in detail. Section 4 shows the performance reports generated by TPCC-UVa in order to meet TPC-C standard requirements. Section 5 shows the use of TPCC-UVa for measuring different aspects of system performance on real machines. Finally, Section 6 concludes the paper.

2 Overview of the TPC-C standard specification

The TPC-C benchmark specification simulates the execution of a mixture of read-only and update intensive transactions that simulate the activities found in complex OLTP application environments [1]. The TPC-C workload is determined by the activity of a set of terminals that request the execution of different database transactions, simulating the business activity of a wholesale supplier.

Five different transaction types are defined by the standard. The *New Order* transaction consists of entering a complete order through a single database transaction; the *Payment* transaction enters a customer's payment; the *Order Status* transaction queries the status of a customer's last order; the *Delivery* transaction processes a batch of ten new, not-yet-delivered orders; finally, the *Stock Level* transactions determines the number of recently sold items that have a stock level below a specified threshold.

When a terminal send the transaction request it waits to receive the results in all cases, except for the Delivery transaction, that simulates a transaction executed in deferred mode. The structure of the corresponding database is composed by several tables, with different characteristics with respect to their scheme and cardinality. This benchmark includes a scalability criteria that allows to simulate a realistic workload, allowing to change the database size and the number of transaction terminals for a more accurate simulation of the machine capabilities.

After the execution of the benchmark during a given period of time, the number of New Order transactions executed per minute gives the performance metric, called *transactions-per-minute-C* (tpmC). The TPC-C benchmark also includes a performance value that takes into account the cost of the system under test, the *price-per-tpmC*, to allow a comparison in terms of price/performance. Additional details can be found in the TPC-C standard specification [1].

3 TPCC-UVa architecture and implementation

The TPCC-UVa implementation is composed by five different modules that collaborate to perform all the necessary activities to measure the performance of the system under test. Figure 1 shows the TPCC-UVa architecture. The modules are the following.

Benchmark controller This module interacts with the user, populating the database and allowing the launch of different experiments.

Remote Terminal Emulator (RTE) There is one RTE process per active terminal in the benchmark execution. It simulates the activity of a remote terminal, according with TPC-C specifications.

Transaction Monitor This module receives all the requests from the RTEs, executing queries to the underlying database system.

Checkpoints controller This module performs checkpoints periodically in the database system, registering timestamps at the beginning and the end of each checkpoint.

Vacuum Controller This module avoids the degradation produced by the continuous flow of operations to the database.

Interprocess communication is carried out using both shared-memory structures and system signals, allowing to run the benchmark in any Unix-like, shared-memory multiprocessor environment. The following subsections describe each module in more detail.

3.1 Benchmark Controller

The Benchmark Controller (BC) allows the user to access the benchmark functionality. It performs the following functions.

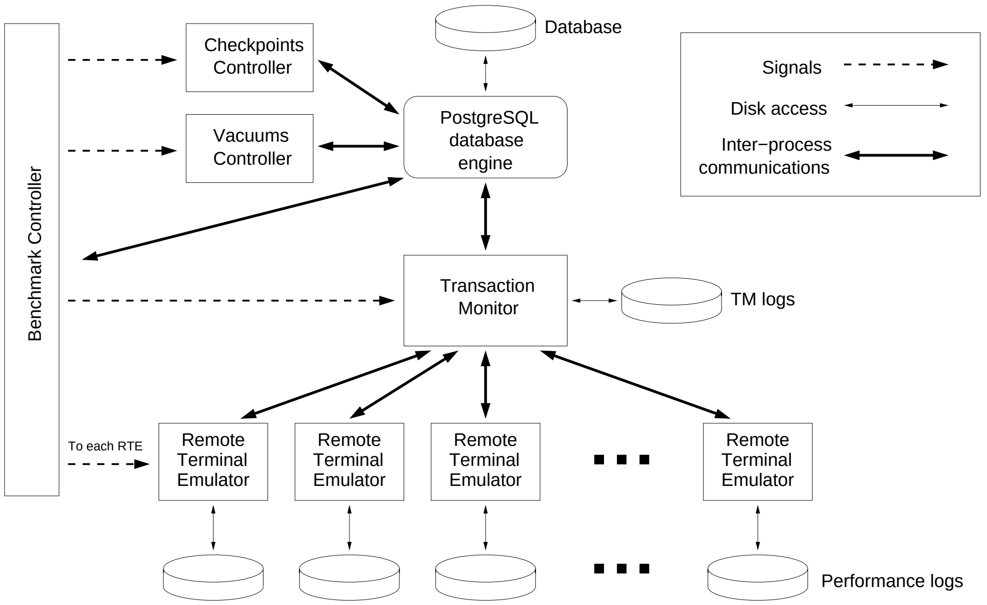


Figure 1: TPCC-UVa architecture.

Database initial population: It creates a new database to run a test. The database is composed by the nine tables defined in the TPC-C specifications, together with their required population and scalability characteristics. Different mechanisms to ensure the reference integrity of the data, such as primary and foreign keys, are also included.

Database consistency check: This option allows the user to check the consistency of the active database, to see if it meets the conditions described by the TPC-C standard to run a test on it.

Restoring an existent database: This option eliminates the modifications performed in the database tables by a previous test run. The purpose of this option is to rebuild a database to run a new test according with the TPC-C requirements without the need of creating a new one from scratch, a time-consuming operation.

Deleting a database: This option allows the user to delete the current database.

Executing a test: This option launches the TPCC-UVa modules that allow to run a measurement test. Such a test is composed by three intervals: the *ramp-up* period, a time when the performance of the system is not stable yet and therefore will not be considered for the performance measurement; the *measurement interval*, where the performance measurement is done; and the *end-of-test period*, when the Benchmark Controller stops all the related processes.

To execute a test, the user should define different execution parameters, such as the number of warehouses to be considered, the ramp-up period, the measurement interval and the configuration of the Vacuum Controller (described in Section 3.5). To run a test, the Benchmark Controller starts the Transaction Monitor, up to ten Remote Terminal Emulators for each one of the selected warehouses, and the Checkpoint and Vacuum Controllers (see Fig. 1). The Benchmark Controller also defines the experiment timings, informing each module about the current interval while executing a test.

Summary results of last test: This option reads and processes the benchmark logs produced by the set of Remote Terminal Emulators and the Transaction Monitor during the execution of the test. The information provided by the logs can be divided in two parts. The first one is the number of New Order transactions executed per minute, together with the response time of the executed transactions. This information will determine the performance of the system under test. The second part is the data needed to ensure that the test has been performed following the TPC-C specifications, such as the terminal response times and the relative percentage of each transaction in the executed transaction set. Both data types should be processed by the Benchmark Controller to ensure that the test is valid and to return the TPCC-UVa Transactions-Per-Minute (tpmC-uva) metric.

The Transactions-Per-Minute metric returned by TPCC-UVa is called tpmC-uva instead of tpmC. The reason is that, as we said in Section 1, the metric obtained with TPCC-UVa should not be compared with tpmC values obtained by approved implementations of the benchmark.

3.2 Remote Terminal Emulators

The Remote Terminal Emulators (RTE from here on) generate the transaction requests for the system. Each RTE runs as an individual process, generating new transactions according with the requirements of the TPC-C benchmark specification. Once the measurement time is expired, the Benchmark Controller stops each one of the RTE using system signals. The RTE capabilities are the following.

User simulation: Each RTE simulates the behavior of a user connected to it, performing transaction type selection and transaction input data generation. It also simulates two related wait times: "keying time" and "think time".

Terminal simulation: Each RTE generates the output required by each terminal, showing the information introduced by the simulated user and the results obtained once the transaction is executed. Although each RTE can show this information in the standard output, the generated output is usually redirected to `/dev/null` to avoid collapsing the system console.

Transactions management: Each RTE generates a transaction type according with the TPC-C specifications, sending it to the Transactions Monitor. If the transaction is interactive, the results are sent back to the corresponding RTE once the transaction is completed.

Transaction response time measurement: Each RTE measures the response time for each one of the transactions requested. This data is stored locally in a log file, together with additional information that will be needed for the performance measurement of the system under test.

3.3 Transactions Monitor

The Transactions Monitor (TM from here on) receives the transaction requests from all the RTEs, passing them to the database engine and returning the generated results back to the RTEs. The transactions are executed according with their arrival order. The TM also registers the results of the delayed execution of the Delivery transaction and, when needed, data related to errors in the execution of transactions. The TM is activated and deactivated by the Benchmark Controller.

Clause 2.3.5 of the TPC-C standard specification [1] indicates that "if transactions are routed or organized within the SUT [System Under Test], a commercially available transaction processing monitor" is required, with a given set of functionalities. To avoid the use of commercially-available software, our TM does not route or organize transactions, but only queues them for execution in arrival order.

3.4 Checkpoints Controller

The Checkpoints Controller is responsible for ordering checkpoints periodically, registering the timestamps at the beginning and end of each checkpoint, according with Clause 5.5.2.2 of the TPC-C standard specification [1]. The first checkpoint is performed when the Checkpoints Controller is activated, at the beginning of the measurement interval.

3.5 Vacuum Controller

The Vacuum Controller mitigates the negative effects of a continuous flow of transaction executions in the database system. This controller is needed because the chosen database engine (PostgreSQL) keeps residual information that may slow down the database operation. To avoid a performance loss in the execution of long tests (i.e. more than two hours), the Vacuum Controller executes periodically the PostgreSQL `vacuum` command [9]. The user can configure the interval between vacuums and their maximum number.

3.6 TPCC-UVa communication procedures

Communication between the Transaction Monitor and each Remote Terminal Emulator is implemented using the communication procedures provided by Unix System V IPC interface, such as semaphores, shared memory and message queues [11]. The communication between the TM and the RTEs is based on the use of a single queue of pending transaction requests. This queue is used by the RTEs to submit transaction requests to the TM. The incoming order of the requests into the TM determine their execution order. A synchronization semaphore is used to manage reads and writes to this queue. Once a transaction is completed, the results are transmitted from the MT to the RTE that issued the request through a shared-memory data structure. Again, a semaphore is used to manage each data structure.

4 TPCC-UVa reports

The execution of TPCC-UVa on a System-Under-Test (SUT from here on) returns different performance metrics and plots. As an example, in this section we will describe in detail the results and plots obtained by TPCC-UVa version 1.2.3 on a dual core multiprocessor system. The SUT

```
Test results accounting performed on 2006-11-30 at 00:00:59 using 30 warehouses.

Start of measurement interval:  20.004883 m
End of measurement interval:  140.004983 m
COMPUTED THROUGHPUT: 367.791 tpmC-uva using 30 warehouses.
101424 Transactions committed.

NEW-ORDER TRANSACTIONS:
44135 Transactions within measurement time (50583 Total).
Percentage:  43.515%
Percentage of "well done" transactions:  97.100%
Response time (min/med/max/90th):  0.006 / 0.813 / 84.781 / 1.240
Percentage of rolled-back transactions:  0.986% .
Average number of items per order:  9.400 .
Percentage of remote items:  1.036% .
Think time (min/avg/max):  0.000 / 12.029 / 120.000

PAYMENT TRANSACTIONS:
44099 Transactions within measurement time (50712 Total).
Percentage:  43.480%
Percentage of "well done" transactions:  97.746%
Response time (min/med/max/90th):  0.002 / 0.596 / 89.864 / 0.960
Percentage of remote transactions:  14.148% .
Percentage of customers selected by C_ID: 39.087% .
Think time (min/avg/max):  0.000 / 11.971 / 120.000

ORDER-STATUS TRANSACTIONS:
4417 Transactions within measurement time (5081 Total).
Percentage:  4.355%
Percentage of "well done" transactions:  97.804%
Response time (min/med/max/90th):  0.001 / 0.703 / 84.308 / 1.080
Percentage of clients chosen by C_ID: 39.280% .
Think time (min/avg/max):  0.000 / 10.028 / 93.000

DELIVERY TRANSACTIONS:
4387 Transactions within measurement time (5057 Total).
Percentage:  4.325%
Percentage of "well done" transactions:  99.544%
Response time (min/med/max/90th):  0.000 / 0.122 / 72.625 / 0.010
Percentage of execution time < 80s :  100.000%
Execution time min/avg/max:  0.019/0.631/75.965
No.  of skipped districts:  0 .
Percentage of skipped districts:  0.000%.
Think time (min/avg/max):  0.000 / 5.013 / 47.000

STOCK-LEVEL TRANSACTIONS:
4386 Transactions within measurement time (5061 Total).
Percentage:  4.324%
Percentage of "well done" transactions:  99.772%
Response time (min/med/max/90th):  0.007 / 0.714 / 76.328 / 1.120
Think time (min/avg/max):  0.000 / 4.999 / 47.000

Longest checkpoints:
Start time Elapsed time since test start (s) Execution time (s)
Thu Nov 30 01:51:30 2006 6630.787000 11.000000
Thu Nov 30 01:21:20 2006 4820.357000 10.400000
Thu Nov 30 02:21:41 2006 8441.789000 10.200000
Thu Nov 30 00:51:10 2006 3010.331000 10.009000

No vacuums executed.

>> TEST PASSED
```

Figure 2: Results summary of a TPCC-UVa benchmark execution on a Dual Core Opteron multiprocessor system.

is a server equipped with two Dual Core AMD Opteron Processor 265 at 1 800 MHz (seen by Linux as four different processors), 2 GBytes of RAM and a RAID-5 storage system, using a LSI Logic MegaRAID Serial ATA 300-8X disk controller. The system runs Gentoo Linux with a 2.6.17 kernel, and we used PostgreSQL 8.1.4 as the underlying database system.

Figure 2 shows the results given by TPCC-UVa for a 2-hours test, using 30 warehouses, a ramp-up period of 20 minutes and no vacuum operation. The most important result is the computed throughput, in this case 367.791 tpmC-uva. To be valid, the test should meet some response time requirements, stated in Clause 5.5.1.5 of the TPC-C benchmark. The last line of the results file shown in Fig. 2 indicates whether these requirements have been met in this particular experiment.

In addition with the result summary given in Fig. 2, the TPCC-UVa implementation returns the data needed to draw the plots defined by the TPC-C standard specification. According to the standard, four different plot families should be generated. The plot families are described below.

Frequency distribution of Response Times Clause 5.6.1 of the TPC-C standard specification requires to build a graph called Response Time Distribution, that shows the number of transactions of each different transaction type that were completed in a given response time. Figure 3 shows both the plot as described by the TPC-C standard specification, and the response time distributions of the five transaction types given by TPCC-UVa.

Response Times vs. Throughput for the New Order transaction Clause 5.6.2 of the TPC-C standard specification requires to build a graph of response times versus throughput for the New Order transaction. The graph must be plotted at approximately 50%, 80% and 100% of the reported throughput rate (additional data points are optional). Figure 4 shows both the plot as described by the TPC-C standard specification, and the corresponding plot obtained with TPCC-UVa.

Frequency distribution of Think Times Clause 5.6.3 of the TPC-C standard specification requires to build a graph with the frequency distribution of Think Times for the New Order transaction. At least 20 different intervals of equal length must be reported. Figure 5 shows both the plot as described by the TPC-C standard specification, and the corresponding plot obtained with TPCC-UVa.

Throughput of the New Order Transaction Clause 5.6.4 of the TPC-C standard specification requires to build a graph with the throughput of the New Order transaction versus elapsed time, for both the ramp-up period and measurement interval. At least 240 different intervals should be used, with a maximum interval size of 30 seconds. The opening and the closing of the measurement interval must also be reported and shown on the graph. Figure 6 shows both the plot as described by the TPC-C standard specification, and the corresponding plot obtained with TPCC-UVa.

5 System performance measurement

In this section we use TPCC-UVa to measure different aspects of system performance. The following sections discuss some results obtained with TPCC-UVa to compare two important aspects of system performance, such as file system performance and multi-core capabilities. The purpose of these experiments is to show how TPCC-UVa can be used to obtain a reliable measure of different systems under test.

5.1 File systems performance comparison

We have used TPCC-UVa to measure the performance of different file system implementations under Linux. The SUT for this experiment is a server equipped with two Dual Core AMD Opteron Processor 265 at 1 800 MHz with 2 GBytes of RAM and running Gentoo Linux with a 2.6.17 kernel. We have used a Seagate Barracuda 7200.7 Serial ATA disk to store the database. This disk was divided into four primary partitions, each one formatted using a different file system type.

The file system types considered in this experiment were the following: ext2fs, the classical file system for Linux [2]; ext3fs, a version of ext2fs with journaling; ReiserFS version 3.6, a journaling file system for Linux based on balance tree algorithms, developed by Namesys; and JFS, based on IBM's journaled file system technology and now open-source.

We have run several experiments using each file system to store the TPCC-UVa database maintained by PostgreSQL. Each experiment ran for two hours, with ramp-up periods of 20 minutes and no vacuum operations. Figure 7 shows the tpmC-uva obtained using different number of warehouses for each one of the four file systems considered.

As can be seen in Fig. 7, ext2fs gives better results than file systems with journaling, particularly ReiserFS (3.0 percent slower) and JFS (8.9 percent slower). The maximum number of warehouses that the SUT was capable to serve was 27 for ext2fs, ext3fs and ReiserFS, while JFS allowed to run the benchmark with at most 25 warehouses.

The response time requirements defined by TPC-C were not met with more warehouses.

5.2 Dual- and single-core performance comparison

Finally, in this section we compare the performance of a SUT with two dual-core processors with respect to an identical system but with two single-core processors.

The dual-core SUT has two Dual Core Opteron 265 (seen by Linux as four processors), while the single-core SUT has two Single Core Opteron 246 processors. Both systems have 2 GBytes of RAM and a RAID-5 storage system, using a LSI Logic MegaRAID Serial ATA 300-8X disk controller and a ReiserFS file system. Both systems run Gentoo Linux with a 2.6.17 kernel. We have run different 2-hours experiments, with ramp-up period of 20 minutes and with no vacuum operations.

Figure 8 shows the tpmC-uva obtained using different number of warehouses for both SUTs, together with the maximum throughput of the New Order transaction in both cases. The results show that the dual-core architecture allowed a workload of up to 30 warehouses, with a 28.1% performance gain over the single-core architecture, that only dealt with up to 25 warehouses. These results show that, as expected, multi-core architectures are a valid choice for running transaction-oriented database workloads.

6 Conclusions

This paper describes TPCC-UVa, an open-source implementation of the TPC-C benchmark intended for measuring performance of parallel and distributed systems. The implementation simulates the execution of an OLTP environment according with the TPC-C standard specification. The major characteristics of the TPC-C specification has been discussed, together with a description of the TPCC-UVa architecture, performance metrics and plots generated and real examples of performance measurements for parallel systems. TPCC-UVa can be freely downloaded from `http://www.infor.uva.es/~diego/tpcc-uva.html`.

Acknowledgments

The author is partially supported by the European Commission under grant RII3-CT-2003-506079, and by Castilla-Leon Regional Government under grant VA031B06. The author would like to thank Julio A. Hernández and Eduardo Hernández for implementing the first version of TPCC-UVa as part of their BSc. thesis; Belén Palop for her help during the development of the benchmark; Javier Ramos for his expertise and support in Linux environments; and Joaquín Adiego, Pablo de la Fuente and the Grinbd Group for allowing the use of their computing facilities to run some of the experiments.

References

[1] TPC benchmark C standard specification, revision 5. Transaction Processing Performance Council. `http://www.tpc.org`. Access date: December 2006.

[2] Card, R., Ts'o, T., and Tweedie, S. Design and implementation of the second extended filesystem. In *Proc. of the First Dutch International Symposium on Linux* (December 1994). ISBN 90-367-0385-9.

[3] Eisenberg, A., and Melton, J. Standards in practice. *SIGMOD Rec. 27*, 3 (1998), 53–58.

[4] Henning, J. L. SPEC CPU2000: Measuring CPU performance in the new millennium. *Computer 33*, 7 (2000), 28–35.

[5] Llanos, D. R., and Palop, B. TPCC-UVa: An Open-Source Implementation of the TPC-C Benchmark. In *PMEO-PDS 06, Proc. of IPDPS 2006 Workshops* (April 2006), IEEE Press. ISBN 1-4244-0054-6.

[6] Magnusson, P. S., Christensson, M., Eskilson, J., Forsgren, D., Hallberg, G., Hgberg, J., Larson, F., Moestedt, A., and Werner, B. Simics: A Full System Simulation Platform. *IEEE Computer* (February 2002), 50–58.

[7] NAS parallel benchmark. Access date: Dec. 2006. `http://science.nas.nasa.gov/Software/NPB`.

[8] Piernas, J., Cortés, T., and García, J. M. Traditional file systems versus DualFS: a performance comparison approach. *IEICE Trans. Inf. and Syst. E87-D*, 7 (July 2004).

[9] PostgreSQL 8.1.4 reference manual. PostgreSQL Global Development Group, 2005.

[10] Rogers, A., Carlisle, M. C., Reppy, J. H., and Hendren, L. J. Supporting dynamic data structures on distributed-memory machines. *ACM Trans. Program. Lang. Syst. 17*, 2 (1995), 233–263.

[11] Stevens, W. R. *Advanced programming in the Unix environment*. Addison-Wesley, 1993. ISBN 0-201-56317-7.

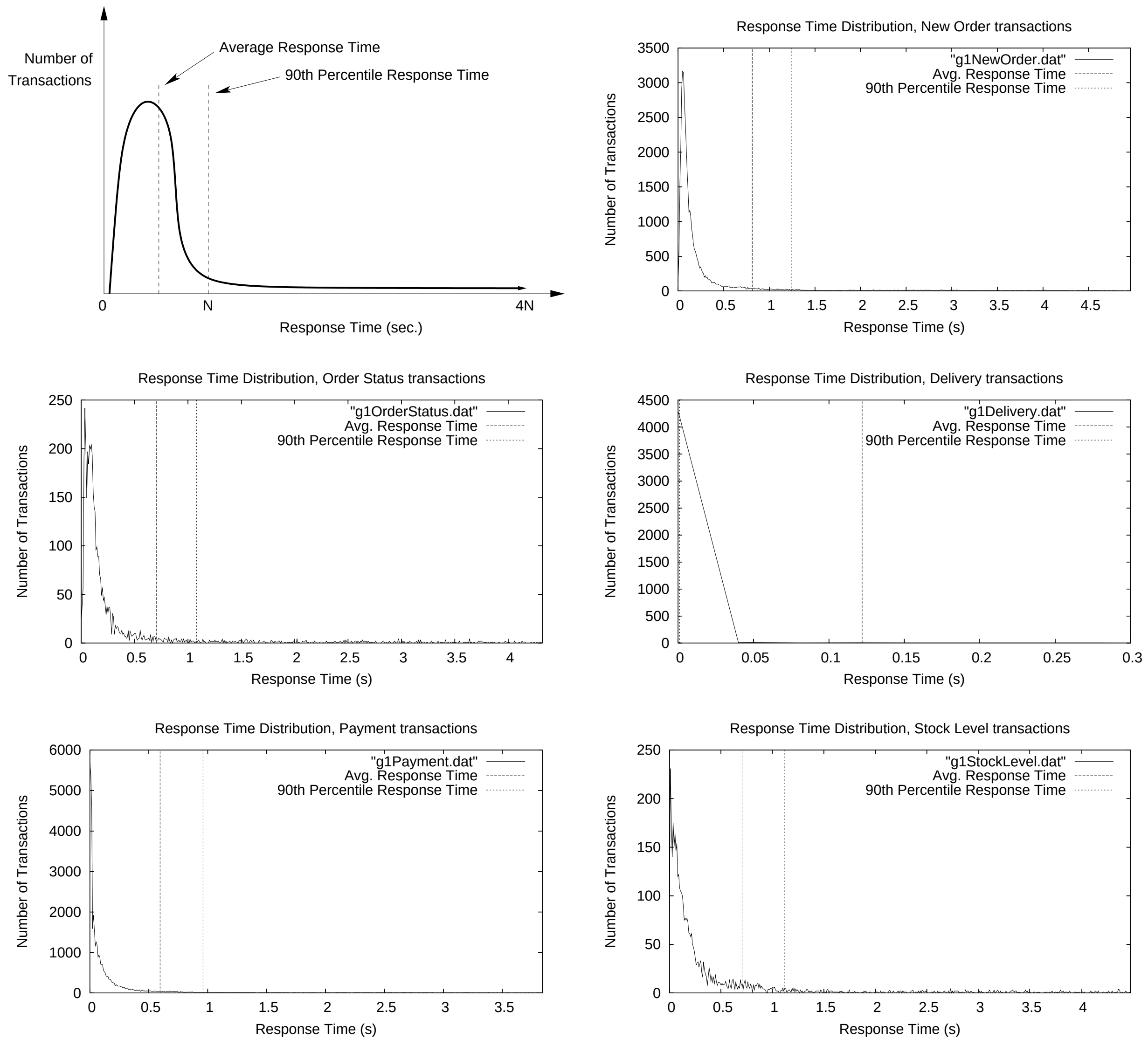


Figure 3: Response time distribution as required by clause 5.6.1 of the TPC-C standard specification (upper left corner) and response time distribution of the five transaction types given by TPCC-UVa for the System Under Test described in Section 4.

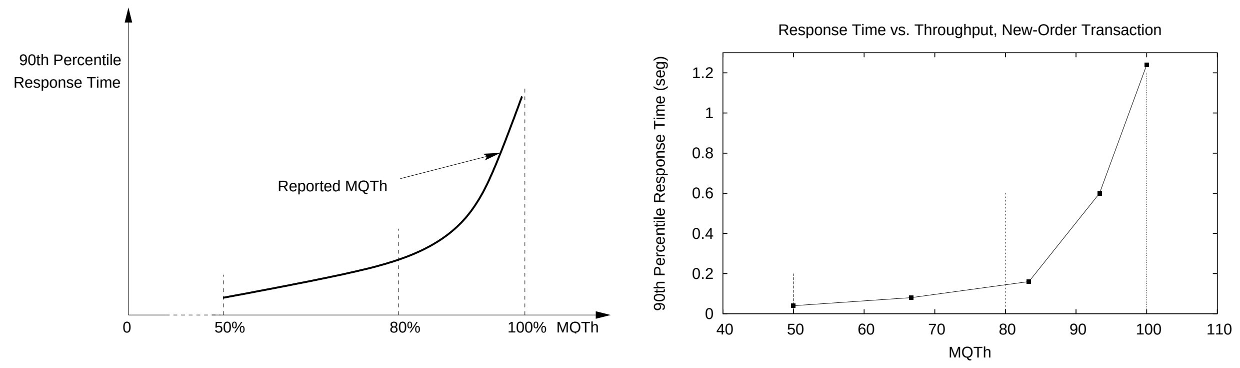


Figure 4: Response times vs. throughput for the New Order transaction as required by clause 5.6.2 of the TPC-C standard specification (left) and the corresponding plot returned by TPCC-UVa for the System Under Test described in Section 4, with configurations of 15, 20, 25, 28 and 30 warehouses (right).

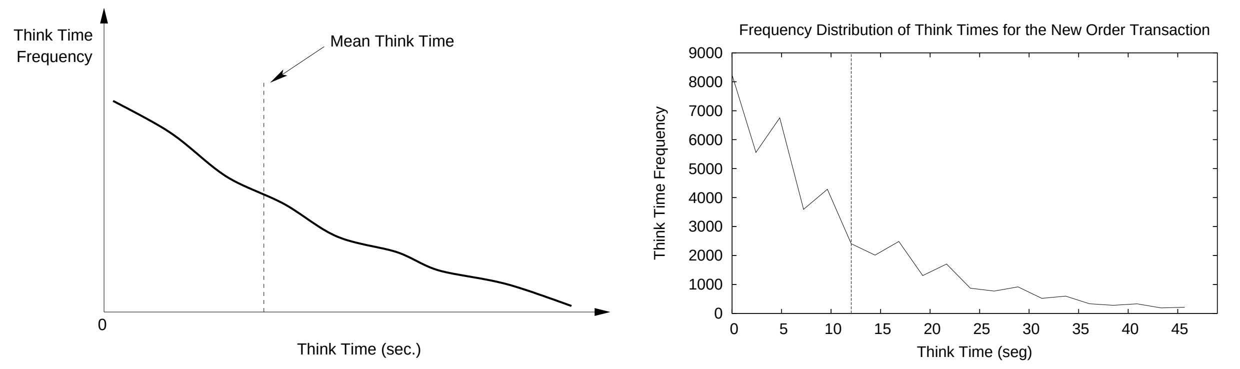


Figure 5: Frequency distribution of Think Times for the New Order transaction as required by clause 5.6.3 of the TPC-C standard specification (left) and the corresponding plot returned by TPCC-UVa for the System Under Test described in Section 4 (right).

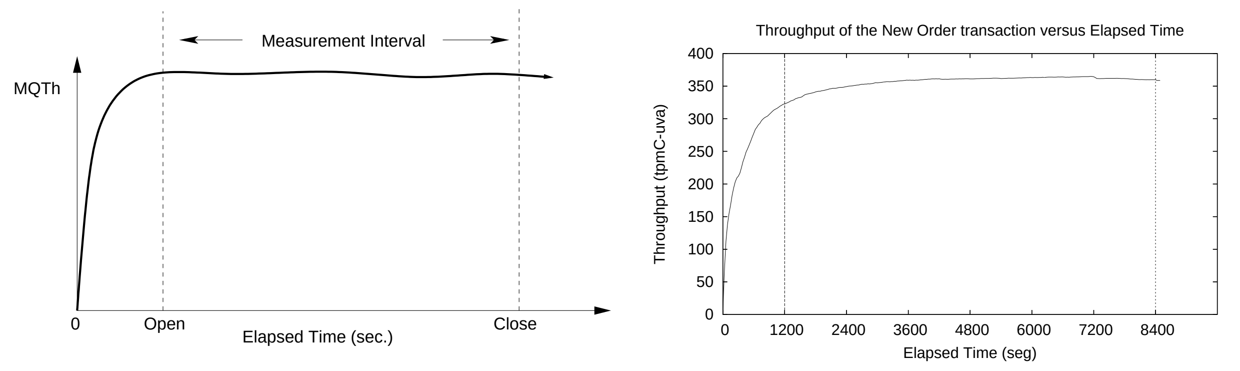
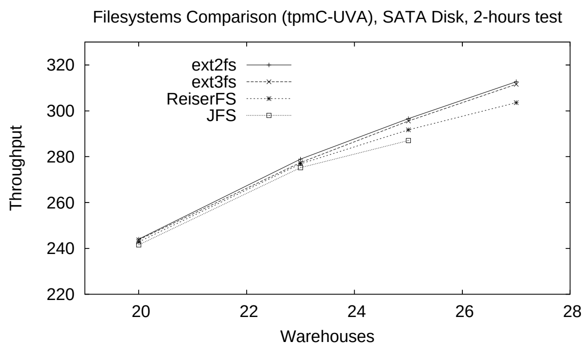


Figure 6: Throughput of the New Order transaction as required by clause 5.6.4 of the TPC-C standard specification (left) and the corresponding plot returned by TPCC-UVa for the System Under Test described in Section 4 (right).



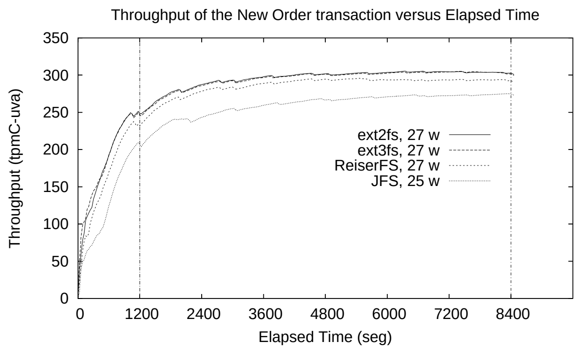
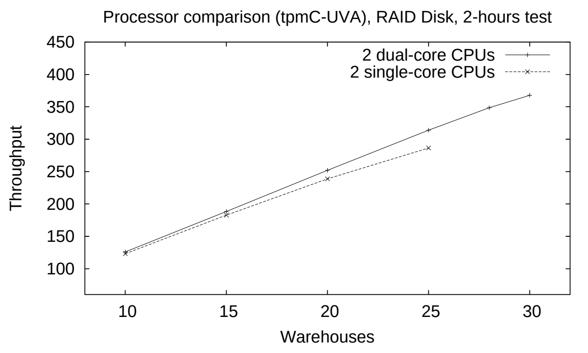


Figure 7: Number of tpmC-uva using different number of warehouses for each file system considered (left) and maximum throughput of the New Order transaction in each case (right).



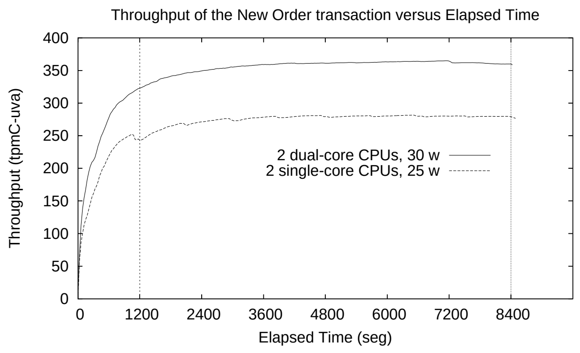


Figure 8: Number of tpmC-uva using different number of warehouses for each SUT considered (left) and maximum throughput of the New Order transaction in each case (right).

XML Search: Languages, INEX and Scoring

Sihem Amer-Yahia
AT&T Labs Research
180 Park Ave.
Florham Park, NJ
sihem@research.att.com

Mounia Lalmas
Queen Mary, University of London
Mile End Road
London E1 4NS
mounia@dcs.qmul.ac.uk

1 Introduction

The development of approaches to access XML content has generated a wealth of issues in information retrieval (IR) and database (DB) (e.g., [2, 15, 17, 20, 19, 47, 26, 32, 24]). While the IR community has traditionally focused on searching unstructured content, and has developed various techniques for ranking query results and evaluating their effectiveness, the DB community has focused on developing query languages and efficient evaluation algorithms for highly structured content. Recent trends in DB and IR research demonstrate a growing interest in merging IR and DB techniques for accessing XML content. Support for a combination of "structured" and full-text search for effectively querying XML documents was unanimous in a recent panel at SIGMOD 2005 [3], and is being widely studied in the IR community [20].

This paper presents an overview of the recent research in XML search, and is composed of 5 sections. Section 2 describes query languages for XML search with an emphasis on the role of document structure in their design and semantics. These languages are mainly inspired from DB work. In IR, many proposals for accessing XML content have been made possible due to INEX which has allowed the evaluation and the comparison of the effectiveness of different XML search approaches. Section 3 gives an overview of INEX, the INitiative for the Evaluation of XML retrieval (INEX) [20]. In Section 4, we describe approaches that have been developed in order to score XML elements according to their relevance. Much of these works were developed and investigated within INEX, but the section also includes work that has been done in DB. Section 5 concludes with a brief summary and discussion of open issues in XML search.

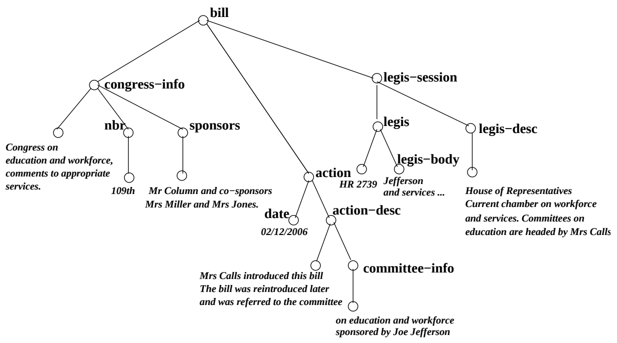

bill
congress−info
nbr
sponsors
Congress on education and workforce, comments to appropriate services.
109th
Mr Column and co−sponsors Mrs Miller and Mrs Jones.
action
date
02/12/2006
action−desc
Mrs Calls introduced this bill The bill was reintroduced later and was referred to the committee
committee−info
on education and workforce sponsored by Joe Jefferson
legis−session
legis
HR 2739
legis−body
Jefferson and services ...
legis−desc
House of Representatives Current chamber on workforce and services. Committees on education are headed by Mrs Calls

Figure 1: Document Excerpt from the LoC

This paper is based on two recent tutorials on XML search [7, 8] and is by no means exhaustive. It is meant as a reference overview for query languages, INEX, and scoring methods for XML, with a strong focus on the latter two. Readers are invited to explore the intricacies of each language, scoring method and INEX in the appropriate references at the end of this paper.

2 Query Languages

Many query languages with varying expressiveness [12] have been proposed for XML search. These languages range from simple keyword search to sophisticated proximity and stemming conditions on keywords combined with complex queries on structure. Document structure plays a major role in the design and semantics of these languages. Even when the language supports Boolean

1

search only, document fragments may be returned, as opposed to whole documents.

We use the example document in Figure 1, extracted from the Library of Congress (LoC) collection [31]. The document describes a bill on education and workforce and exhibits a mix of structured and text information.

In XML search, document structure is used to determine, on behalf of users, which document fragments are more meaningful to return as query answers. It is also used to specify query conditions on structure that limit the search context to specific XML elements, as opposed to whole documents. We classify existing languages for XML search according to their use of conditions on document structure in the query.

Keyword-Only Queries. Languages in this family are a natural evolution from traditional IR languages and are expressed as a conjunction of keywords. The main characteristic of these languages is their ability to return XML fragments. Examples of such languages are found in XRANK [24], nearest-neighbor queries [42], XKSEARCH [55], and NEXI content-only queries [48] developed by INEX [20] (See Section 3). The most notorious method in DB, is to compute lowest common ancestors (LCAs) of query keywords. For example, the query `''Jefferson'' && ''workforce''` applied to the document in Figure 1, would return `<committee-info>` and `<legis-session>` elements as query answers. Variants of the LCAs include meaningful LCAs as in [30] and in XIRQL [19], which are pre-determined XML fragments. Other methods, used in IR, are described in Section 4.

Tag and Keyword Queries. This family of languages includes XSEarch [17] which allows to annotate keywords in the query with tags. For example, the query above could be enhanced as follows `committee-info:Jefferson workforce`, in which case only `<committee-info>` will be returned. XSEarch may also consider semantically related tags as query results.

Path and Keyword Queries. This family of languages exhibits more sophisticated conditions on structure ala XPath [53]. It includes XPath 2.0 [53], XIRQL [19], XXL [47], and NEXI Content-And-Structure (CAS) queries [48] (See Section 3). While the exact syntax of each language is not of interest here, they all agree on an existential interpretation of the path conditions.

XQuery and Keyword Queries. Languages in this family combine the full power of XQuery [52] and a range of *full-text* predicates (as opposed to simple Boolean predicates). Schema-free XQuery offers the ability to specify LCAs with XQuery. For example, the query below returns a bill if there exists a meaningful LCA of `<legis>` and `<legis-desc>` elements, in this case `<legis-session>`, containing the keywords `''Jefferson''` and `''workforce''` respectively. A meaningful LCA is defined as an LCA which makes sense in the context of a given application. For example, returning footnote or paragraph elements may be too fine a granularity to the user.

```
for $b in //bills/bill
    $tmp1 in $b//legis
    $tmp2 in $b//legis-desc
where fn:contains($tmp1, 'Jefferson')
 and  fn:contains($tmp2, 'workforce')
      and exists mlcas($tmp1, $tmp2)
return <result> {$b} </result>
```

TeXQuery [2] and its W3C successor, XQuery Full-Text [54], extend XQuery with composable full-text search predicates such as proximity distance and stemming. The query below written in XQuery Full-Text, returns newly constructed elements containing the number and legislation description of bills produced after January 12th, 2006 and which contain `''Jefferson''` and `''workforce''` within a window of 5 words and using stemming on `''workforce''` if necessary.

```
for $b in //bills/bill
where $b ftcontains 'Jefferson' &&
       'workforce' with stemming
       window 5 words and
       $b//date > 01/12/2006
return <res> {$b//nbr,
              $b//legis-desc} </res>
```

3 INEX Evaluation

There has been many XML search systems that have been developed to implement the query languages described in Section 2, many of which being evaluated within the INEX campaign. In this section, we describe how INEX is providing methods to evaluate the effectiveness of query results.

Evaluating how effective XML search systems are, requires test-beds where the evaluation paradigms are provided according to criteria that take into account the imposed structural aspects. In March 2002, the INitiative for the Evaluation of XML retrieval (INEX) [20] started to address these issues. The aim of the INEX initiative is to establish an infrastructure and to provide means, in the form of a large test collection and appropriate scoring methods, for evaluating how effective are content-oriented XML search systems (as opposed to structure-oriented systems).

Evaluating retrieval effectiveness is typically done by using test collections assembled specifically for evaluating particular retrieval tasks. A test collection consists of a set of documents, a set of user requests (the so-called topics, or queries) and relevance assessments of the documents with respect to the queries. Most of existing test collections treat documents as atomic units and make assumptions that become invalid in the context of XML retrieval, which is mainly due to the lack of a predefined uniform unit of retrieval and the dependency that exists among retrievable components. As such, the characteristics of traditional test collections have been adjusted in order to appropriately evaluate content-oriented XML retrieval effectiveness.

Document Collection The INEX corpus is composed of the full-texts, marked up in XML, of 16.819 articles of the IEEE Computer Society's publications from magazines and transactions, covering the period of 1995-2004, and totaling 735 megabytes in size. On average an article contains 1,532 XML nodes, where the average depth of a node is 6.9. The overall structure of a typical article consists of a frontmatter (containing e.g. title, author, publication information and abstract), a body (consisting of e.g. sections, sub-sections, sub-sub-sections, paragraphs, tables, figures, lists, citations) and a backmatter (including bibliography and author information).

Topics INEX defined two types of topics: *Content-only (CO)* queries which are similar to those used in TREC and, *Content-and-structure (CAS)* queries which contain structural constraints that can refer to where to look for the relevant elements (i.e. support elements) and what types of elements to retrieve (i.e. target elements). NEXI [48] is the query language defined for this purpose and is based on XPath [53]. It should be noted that NEXI is more focused on querying content (content-oriented retrieval) than many of the languages presented in Section 2. As in TREC, an INEX topic consists of the standard title, description and narrative fields. Up to date, the INEX has accumulated a total of 268 topics, most of them with associated relevance assessments.

Tasks The main INEX activity is the ad-hoc retrieval task. Ad-hoc retrieval in IR is described as a simulation of how a library might be used, and it involves the searching of a static set of documents using a new set of topics. Two ad hoc retrieval sub-tasks have been identified since 2002, depending on how structural constraints are expressed. In the CO sub-task, using CO queries, it is left to the retrieval system to identify the most appropriate XML elements to return to the user. Depending on how we assume a user wants the output of an XML retrieval system to be, three different strategies have been defined in 2005. In a focused strategy, we assume that a user prefers a single element that most exhaustively discusses the topic of the query, while at the same time it is most specific to that topic. In a thorough strategy, we assume that a user prefers all highly exhaustive and specific elements, and in a fetch and browse strategy we assume that a user is interested in highly exhaustive and specific elements that are contained only within highly relevant articles. In the CAS sub-task, using CAS topics, structural constraints can be interpreted as strict or vague (they are viewed as hints). Strict and vague interpretation can be applied to both support and target elements, giving a total of four strategies for this sub-task.

Relevance XML elements forming a document can be nested. Some elements are large (e.g., sections) and others small (e.g., paragraphs). Since retrieved elements can be

3

at any level of granularity, an element (the larger element) and one of its child elements (the smaller element) can both be relevant to a given query, but the child element may be more focused to that given query than its parent element and may thus be a better element to retrieve [38].

INEX uses two graded dimensions to express relevance: exhaustivity and specificity. Exhaustivity is defined as a measure of how exhaustively an XML element discusses the topic of request, while specificity is defined as a measure of how focused the element is on the topic of request (i.e. discusses no other, irrelevant topics). Exhaustivity refers to the standard relevance criterion used in IR, whereas specificity provides a measure with respect to the size of a component as it measures the ratio of relevant to non-relevant content within an element. The combination of the two dimensions is used to identify those relevant XML elements, which are both exhaustive and specific to the topic of request and hence represent the most appropriate unit of information to return to the user.

Metrics Since its launch in 2002, INEX has been challenged by the issue of how to measure an XML information access system's effectiveness. Up to 2004, INEX used a metric based on the measure of precall, which computes the probability that an XML element viewed by the user is relevant. That is, it interprets precision as the probability that an XML element viewed by a user is relevant at standard recall values. To apply the metric, the two relevance dimensions are mapped to a single relevance scale using quantisation functions. For example, a strict quantisation function is used to evaluate retrieval methods with respect to their capability of retrieving highly relevant elements. A generalised function is used to credit retrieved elements according to their degree of relevance, thus also allowing to reward less relevant elements.

The latter is important as it allows considering near-misses when calculating effectiveness performance. Due to the lack of an atomic predefined unit of retrieval as well as the increased richness of the user's interaction with the system (i.e., browsing), users have access to other, structurally related components from a returned result element. Near-misses are elements, which may be themselves not exactly relevant to the user's query, but from where users can access relevant content. The idea is that XML retrieval approaches are partially rewarded for finding such elements, as it is still better to return near-misses than irrelevant elements.

The metric, in its current version, cannot consider overlapping results. This is because, in INEX, the recall-base (the set of relevant elements for each given query) consists of a large proportion of overlapping elements (if an element is relevant, so is its parent element). This so-called overpopulated recall-base can lead to misleading effectiveness results because the recall-base contains more relevant elements than an ideal system should in fact retrieve. In fact, perfect recall can only be reached by systems that return all the relevant elements of the recall-base, *including* all the overlapping elements [29].

Therefore, INEX adopted in 2005 the eXtended Cumulated Gain metric, XCG [28], which considers both system and user-oriented evaluation aspects. User-oriented measures allow to reason about a system's ability to satisfy users, and typically focus on the early ranks of a system's output as users are more likely to limit their search to these results. System-oriented measures allow system developers to obtain an overall picture of performance.

4 Scoring

XML search systems aim at providing more precise access to XML documents by retrieving document components (the so-called XML elements) instead of whole documents in response to users' queries. This was illustrated in the description of INEX and non-INEX XML search languages in Section 2. In those languages, XML elements of any granularity (e.g., a paragraph or the section enclosing it) are potential answers to a query, as long as they are relevant. This constitutes a major departure from traditional IR: XML retrieval systems need not only score elements with respect to their relevance to a query, but also determine the appropriate level of component granularity to return to users. In this section, we discuss the main challenges associated with scoring and returning elements at the right level of granularity. We also briefly illustrate how these challenges have been or are currently being addressed in INEX.

Term and element statistics Classical scoring methods in IR make use of term and document statistics, the most common ones being term frequency, tf, and document frequency df. tf is the number of occurrences of a term in a document and df is the number of documents in which the term appears. Scoring at element level requires statistics at element level. It is not straightforward to simply replace document by element (etf and ef), since elements in XML documents are nested. Suppose that a section element is composed of two paragraph elements. How should we compute the etf and ef values of a term in the paragraphs and the section?

One approach in [33] is to build an index for each element type (e.g., article, section, paragraph, etc), and compute the statistics for each element type separately. A query is then ran in parallel on each index (e.g., using any or several scoring models). The scores in each result set are normalized, so that scores from different indices are comparable. Finally, results are merged into a single result set. This avoids problems arising from nested elements as elements of different types are treated separately. A different approach is to compute etf and ef statistics for leaf-elements only, which are then used to score the leaf-elements themselves. All non-leaf elements are scored based on some (weighted) combination of the score of their children elements. The propagation of score starts from the leaf-elements and can consider the distance between the element being considered and its descendant leaf-elements [22, 40].

Many approaches ignore element nesting by simply calculating ef with respect to all elements of the collection [43], or partly consider it by estimating ef across elements of the same type [45] or using df [16]. etf is often calculated based on the concatenation of the text of the element and that of its descendants [36, 45].

It is not yet clear what works best, as obviously which approach to follow would depend on the collection, the types of elements (i.e., the DTD) and their relationships, and the scoring method. An interesting research would be to investigate all developed term and element statistics approaches within a uniform and controllable environment to determine those leading to the best performance.

Obtaining consistent and meaningful statistics for effective retrieval can be viewed as the more general problem of the *combination of evidence*. In XML retrieval, term and element statistics are not enough; element size has been shown to be important [44, 35]. The next paragraphs describe other types of evidence.

Structure Statistics Not all elements are equally likely to be appreciated as satisfactory answers to an information need. The structure in XML documents should be used to decide which elements are likely to trigger higher user satisfaction. This is usually done by computing some statistics on the structure itself. One approach [33, 16] is to only index and/or retrieve those element types with the highest distribution of relevant elements in past relevance assessment sets. In INEX, selected types included section, abstract, sub-section, paragraph element type. A different approach [43] is to index only those element types whose average length was above a given threshold. A related strategy is to discard at retrieval all elements smaller than a given threshold (usually expressed in terms of number of words). Both strategies strategy will indeed remove elements of a particular type, which are often considered to be too small to act as a meaningful retrieval unit. [40] however recently showed, that although the so-called small elements should not be returned, they might still influence the scoring of enclosing elements, so they should still be indexed.

Relationships Statistics XML documents are not just documents that are made of components of various types. There is a relationship between the elements, as provided by the logical structure of the XML mark-up. An element, unless it is a root element, has a parent element, which itself may have a parent element. Similarly, non-leaf elements have children elements, and so on. It makes sense to exploit element relationships to score elements.

Some approaches, like those based on the Bayesian network or hierarchical language modeling [50, 36] explicitly capture element relationships. They can associate, explicitly or implicitly, statistics to structural relationships through learning to capture the strength of those relationships. These models need extensive training, which is computationally expensive. A simplified hierarchical language model that reduces the parameter estimation complexity was proposed in [37].

Other approaches propagate terms statistics along the structure, where it is then possible to weight the propagation depending on the relationships [22, 40]. The weighting parameters are usually empirically determined.

A particular relationship is that of the element and its root element. For instance, in INEX, the root element of any element is the article element. Considering this particular relationship has often shown to improve performance (e.g. pivot in [34], root-based contextutalitaion [9], article level language model [43]).

Overlapping Elements It is one task to provide a score expressing how relevant an element is to a query and a different task to decide, from several overlapping relevant elements, which one to return as the *best* answer. For instance, returning a paragraph and its enclosing section should be avoided to ensure that users do not get to see the same information several times. Deciding which element to return obviously depends on the application and its user models. For instance, in INEX, the best element is one that is highly relevant, but also specific to the topic (i.e., does not discuss unrelated topic).

Three types of approaches have been applied within INEX to remove overlap. A common one is to select the highest scored element from each path and return that element [40, 22]. Additional filtering can be applied such as discarding all elements of a given type (e.g., article elements even if they are highly scored [50]). Although the tree structure of a document may have been considered at indexing and/or retrieval time, it is ignored when removing overlap, as elements are treated independently.

An approach that does not treat elements independently when removing overlap is [16], where elements are reranked by iteratively adjusting the score of those elements contained in or containing higher ranked elements in the result list. Overlap is controlled by re-weighting terms occurring in the higher ranked elements to reflect their reduced importance. A second approach [34] looks at the distribution of relevant elements in the tree structure in addition to their score. For instance, an element that has many of its descendants retrieved (overlapping elements), but which are evenly distributed in the corresponding tree structure, and in addition have a similar score to the parent element, would be selected as the best element; otherwise its descendants would be considered instead. This approach was shown to be better than when selecting the highest scored element from each path as best elements.

A third type of approaches argue that the highest ranked elements may not necessarily be the most useful units of retrieval. For instance, a small but highly scored element may not be that useful. [35] ranks elements using a utility function that is based not only on the relevance score of an element, but its size, and the amount of irrelevant information contained in its children elements. An element whose utility value is higher to that of the sum of the utility value of its children is selected as best element. otherwise, the children elements whose utility values are above some threshold are selected.

Structure Constraints Since the beginning of INEX, there has been a debate on how to interpret structural constraints [49]. Initially, INEX required those constraints to be strictly matched. However, specifying an information need is not an easy task, in particular for semi-structured data with a wide variety of tag names. Although users may think they have a clear idea of the structural properties of the collection, there are likely to be aspects to which they are unaware. Obviously, this is very dependent on the actual application. Because of this, a vague interpretation of the structural constraints is followed: an element is relevant if it satisfies the information need, irrespective of the structural constraints, which are mainly considered as structural hints. That is, the aim of XML search system is to return components that contain the information sought after by the user even if the result elements do not exactly meet the structural conditions expressed in the query.

A common approach in INEX has been to manually build a dictionary of tag synonyms (e.g. "p" and "ip1" are considered equivalent) [34, 40]. [35] uses past relevance assessments for CAS queries to automatically build the dictionary. When a CAS topic asks for "section" element, all types of elements assessed relevant for that topic are added to the "section" synonym list. Other approaches decide to ignore the structure constraints in support elements, target elements, or both [9].

Scoring according to structural constraints has also been considered. In [22, 45], elements are first scored on how well the content constraints are satisfied, and the scores are boosted on the basis of matching the structural constraints. There have also been approaches in DB for relaxing structural constraints in XPath (e.g., [6, 41]). The framework in [6] expands query answers by incorporating approximate answers obtained using simple query relaxations. Such relaxations include tag relaxation as in INEX, making a child constraint a descendant constraint, making an element in the query optional and, promoting content conditions to ancestor elements in the query. The framework also defines properties of scoring methods which guarantee that the score of an approximate answer is no higher than the score of an exact answer to a query.

5 Conclusion

We presented an overview of the challenges of designing query languages and scoring methods for XML search and a description of the INEX [20] effort, an initiative for the evaluation of XML retrieval methods. A major challenge for XML search is the combination of the effective retrieval of XML document components from the IR community with efficient query evaluation from the DB community. This challenge calls for a tighter integration of indices on both structure and content and acformalization of queries, in terms of a score-aware algebra, for the joint optimization of queries on both structure and text. An initial effort in this direction can be found in [12] and algorithms for the efficient evaluation of queries on both text and structure can be found in [4, 5, 46]. A second major challenge is the provision of a uniform and controllable platform to study all proposed scoring approaches and their various parameters, as they are so diverse (due to the richness of the XML mark-up and expected user behaviors) so that to obtain fundamental results regarding the best practices in XML scoring.

References

[1] S. Al-Khalifa, C. Yu, H. V. Jagadish. Querying Structured Text in an XML Database. SIGMOD 2003.

[2] S. Amer-Yahia, C. Botev, J. Shanmugasundaram. TeXQuery: A Full-Text Search Extension to XQuery. WWW 2004.

[3] S. Amer-Yahia, P. Case, T. Rölleke, J. Shanmugasundaram, G. Weikum Report on the DB/IR Panel at SIGMOD 2005. http://www.sigmod.org/sigmod/record/issues/0512/p71-column-cooper-1.pdf

[4] S. Amer-Yahia, E. Curtmola, A. Deutsch. Efficient XML Search with Complex Full-Text Predicates. SIGMOD 2006 (To appear).

[5] S. Amer-Yahia, N. Koudas, A. Marian, D. Srivastava, D. Toman. Content and Structure Scoring for XML. VLDB 2005.

[6] S. Amer-Yahia, L. Lakshmanan, S. Pandit. FleXPath: Flexible Structure and Full-Text Querying for XML. SIGMOD 2004.

[7] S. Amer-Yahia, M. Lalmas. Accessing XML Content: From DB and IR Perspectives (Tutorial). CIKM 2005.

[8] S. Amer-Yahia, J. Shanmugasundaram. XML Full-Text Search: Challenges and Opportunities (Tutorial). VLDB 2005.

[9] P. Arvola, J. Kekalainen, M. Junkkari. Query Evaluation with Structural Indices. INEX 2005.

[10] A. Balmin, V. Hristidis, N. Koudas, Y. Papakonstantinou, D. Srivastava, T. Wang. A System for Keyword Proximity Search on XML Databases. VLDB 2003.

[11] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, S. Sudarshan. Keyword Searching and Browsing in Databases using BANKS. ICDE 2002.

[12] C. Botev, S. Amer-Yahia, J. Shanmugasundaram. Expressiveness and Performance of Full-Text Search. EDBT 2006.

[13] J. M. Bremer, M. Gertz. XQuery/IR: Integrating XML Document and Data Retrieval. WebDB 2002.

[14] D. Carmel, Y. S. Maarek, M. Mandelbrod, Y. Mass, A. Soffer. Searching XML Documents via XML Fragments. SIGIR 2003.

[15] T. T. Chinenyanga, N. Kushmerick. Expressive and Efficient Ranked Querying of XML Data. WebDB 2001.

[16] C. Clarke. Controlling Overlap in Content-Oriented XML Retrieval. SIGIR 2005.

[17] S. Cohen, J. Mamou. Y. Kanza, Y. Sagiv. XSEarch: A Semantic Search Engine for XML. VLDB 2003.

[18] M. P. Consens, T. Milo. Algebras for Querying Text Regions: Expressive Power and Optimization. J. Comput. Syst. Sci. 57(3): 272-288 (1998)

[19] N. Fuhr, K. Grossjohann. XIRQL: An Extension of XQL for Information Retrieval. SIGIR 2000.

[20] N. Fuhr, M. Lalmas, Saadia Malik, Z. Szlávik. Advances in XML Information Retrieval. Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004

[21] N. Fuhr, T. Rölleke. A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems. ACM TOIS 15(1), 1997.

[22] S. Geva. GPX - Gardens Point XML IR at INEX 2005. INEX 2005.

[23] T. Grabs, H. Schek ETH Zürich at INEX: Flexible Information Retrieval from XML with PowerDB-XML. INEX Workshop 2002.

[24] L. Guo, F. Shao, C. Botev, J. Shanmugasundaram. XRANK: Ranked Keyword Search over XML Documents. SIGMOD 2003.

[25] V. Hristidis, L. Gravano, Y. Papakonstantinou. Efficient IR-Style Keyword Search over Relational Databases. VLDB 2003.

[26] J. Kamps, M. de Rijke, and B. Sigurbjornsson. Length normalization in xml retrieval. SIGIR 2004

[27] R. Kaushik, R. Krishnamurthy, J. F. Naughton, R. Ramakrishnan. On the Integration of Structure Indexes and Inverted Lists. ICDE 2004.

[28] G. Kazai, M. Lalmas INEX 2005 Evaluation Metrics. INEX 2005

[29] G. Kazai, M. Lalmas, A. P. de Vries. The Overlap Problem in Content-Oriented XML Retrieval Evaluation. SIGIR 2004

[30] Y. Li, C. Yu, H. V. Jagadish. Schema-Free XQuery. VLDB 2004.

[31] Library of Congress. http://lcweb.loc.gov/crsinfo/xml/.

[32] S. Liu, Q. Zou, and W. W. Chu. Configurable indexing and ranking for xml information retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 88–95, 2004.

[33] Y. Mass, M. Mandelbrod. Retrieving the most relevant XML Components. INEX 2004.

[34] Y. Mass, M. Mandelbrod. Using the INEX environment as a test bed for various user models for XML Retrieval. INEX 2005.

[35] V. Mihajlovic, G. Ramirez, T. Westerveld, D. Hiemstra, H. E. Blok, A. P. de Vries. TIJAH Scratches INEX 2005: Vague Element Selection, Image Search, Overlap, and Relevance Feedback. INEX 2005.

[36] P. Ogilvie, J. Callan. Hierarchical Language Models for XML Component Retrieval. INEX 2004.

[37] P. Ogilvie, J. Callan. Parameter Estimation for a Simple Hierarchical Generative Model for XML Retrieval. INEX 2005.

[38] B. Piwowarski, M. Lalmas. Providing Consistent and Exhaustive Relevance Assessments for XML Retrieval Evaluation. CIKM 2004.

[39] A. Salminen, F. Tompa. PAT Expressions: an Algebra for Text Search. Acta Linguistica Hungar. 41 (1-4), 1992

[40] K. Sauvagnat, L. Hlaoua, M. Boughanem XFIRM at INEX 2005: ad-hoc and relevance feedback tracks INEX 2005.

[41] T. Schlieder. Similarity Search in XML Data using Cost-Based Query Transformations. ACM SIGMOD 2001 Web and Databases Workshop.

[42] A. Schmidt, M. Kersten, M. Windhouwer. Querying XML Documents Made Easy: Nearest Concept Queries. ICDE 2001.

[43] B. Sigurbjörnsson, J. Kamps, M. de Rijke The Effect of Structured Queries and Selective Indexing on XML Retrieval. INEX 2005.

[44] B. Sigurbjörnsson, J. Kamps, M. de Rijke The Importance of Length Normalization for XML Retrieval. Journal of Information Retrieval, Vol. 8, Nbr 4, 2005.

[45] M. Theobald, R. Schenkel, G. Weikum. TopX & XXL at INEX 2005. INEX 2005.

[46] M. Theobald, R. Schenkel, G. Weikum. An Efficient and Versatile Query Engine for TopX Search. VLDB 2005.

[47] A. Theobald, G. Weikum. The Index-Based XXL Search Engine for Querying XML Data with Relevance Ranking. EDBT 2002.

[48] A. Trotman and B. Sigurbjörnsson. NEXI, Now and Next. INEX 2004.

[49] A. Trotman, M. Lalmas. The Interpretation of CAS. INEX 2005.

[50] J.-N. Vittaut, P. Gallinari. Machine Learning Ranking and INEX 05. INEX 2005.

[51] J.N. Vittaut, B. Piwowarski, P. Gallinari. An Algebra for Structured Queries in Bayesian Networks. INEX 2004.

[52] The World Wide Web Consortium. XQuery 1.0: An XML Query Language. W3C Working Draft. http://www.w3.org/TR/xquery/.

[53] The World Wide Web Consortium. XML Path Language (XPath) 2.0. W3C Working Draft. http://www.w3.org/TR/xpath20/.

[54] The World Wide Web Consortium. XQuery 1.0 and XPath 2.0 Full-Text. Working draft. http://www.w3.org/TR/xquery-full-text/.

[55] Y. Xu, Y. Papakonstantinou. Efficient Keyword Search for Smallest LCAs in XML Databases. SIGMOD 2005.

[56] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, G. Lohman. On Supporting Containment Queries in Relational Database Management Systems. SIGMOD 2001.

Relaxed Space Bounding for Moving Objects: A Case for the Buddy Tree

[1]Shuqiao Guo [1]Zhiyong Huang [2]H. V. Jagadish [1]Beng Chin Ooi [1]Zhenjie Zhang

[1]Department of Computer Science
National University of Singapore, Singapore
{guoshuqi, huangzy, ooibc, zhenjie}@comp.nus.edu.sg

[2]Department of EECS
University of Michigan, USA
jag@eecs.umich.edu

ABSTRACT

Rapid advancements in positioning systems and wireless communications enable accurate tracking of continuously moving objects. This development poses new challenges to database technology since maintaining up-to-date information regarding the location of moving objects incurs an enormous amount of updates. There have been many efforts to address these challenges, most of which depend on the use of a minimum bounding rectangle (MBR) in a multi-dimensional index structure such as R-tree. The maintenance of MBRs causes lock contention and association of moving speeds with the MBRs cause large overlap between them. This problem becomes more severe as the number of concurrent operations increases. In this paper, we propose a "new" simple variant of the Buddy-tree, in which we enlarge the query rectangle to account for object movement rather than use an enlarged MBR. The result is not only elegant, but also efficient, particularly in terms of lock contention. An extensive experimental study was conducted and the results show that our proposed structure outperforms existing structures by a wide margin.

1. INTRODUCTION

Rapid advancements in positioning systems, sensing technologies, and wireless communications, make it possible to track accurately the movement of thousands of mobile objects. This has led to an urgent need to develop techniques of efficient storage and retrieval of moving objects. Indeed, this topic has received significant interest in recent years [8, 12, 7, 15, 13].

Mobile objects move in (typically two or three-dimensional) space. As such, traditional index techniques for multidimensional data are a natural foundation upon which to devise an index for moving objects. A standard technique for indexing objects with spatial extent is to create a *minimum bounding rectangle* (MBR) around the object, and then to index the MBR rather than the object itself. Many multidimensional index structures, including in particular R-tree and its derivatives [5, 1], follow such an approach.

Moving objects, even if they are modeled as points, are in different locations in space at different times. In an index valid over some period of time, if we wish to make sure to locate the moving object, we can do so by means of a bounding rectangle around the locations of the object within this period of time. (Most spatio-temporal indices also have explicit notions of object velocity, and make linear, or more sophisticated, extrapolations on object position as a function of time. But an MBR is still required to make sure that a search query does not suffer a false dismissal). See our discussion of the popular TPR-tree structure [13] in Section 2.3 for more detail.

The key observation we make is that these MBRs can become quite large, particularly as objects have high velocity or as index structures are used to model the world for longer periods of time. Because of this, too many false dismissals may be observed. If an R-tree like structure is used to hierarchically organize these MBRs, we may find large overlaps between non-leaf MBRs, causing search to proceed on multiple paths down the tree.

A fix to the above problem is to have the tree reflect only a very short period of time. (This is actually a fix only with respect to queries regarding current position. Reducing the time interval of index validity actually makes more difficult predictive queries dealing with future positions of objects.) But this means objects have to be removed and inserted from the tree very frequently, as soon as they move more than a little. Traditional multidimensional index structures have not been designed to support such high update rates. There is always the possibility of a node split upon an insertion and this could back up all the way to the root, requiring conservative choices in concurrency control.

In this paper, we attempt to address these difficulties by redefining the problem of indexing mobile objects. Instead of embedding the velocity information within the index, we attempt to capture it in the query. Now, instead of point objects ballooning into large MBRs, we will have point queries being turned into rectangular range queries. On the surface, this appears to make no difference in terms of performance – so one wonders why bother to make this equivalence transformation?

It turns out that the benefit we get is that we can now build much simpler indices – we only need to consider static objects rather than mobile objects. So we can choose a multidimensional structure with good update properties. In particular, we propose a simple indexing structure based on the Buddy-tree [14] – the Buddy*-tree. The bounding rectangles in the internal nodes are not minimum, and are based on the pre-partitioned cells. This turns the Buddy-tree to a multi-level like grid file [11], in that the union of the lower level bounding spaces span the bounding space of the parent.

To allow concurrent modifications, we adapt the concurrency control mechanism of the R-link tree [9]. Since the Buddy*-tree is a space partitioning-based method, it does not suffer from the high-update cost of the R-tree, and due to the decoupling of velocity information from bounding rectangles, it does not suffer from the overlap problem of the TPR-tree. Experimental studies were conducted, and the results that the Buddy*-tree is much more efficient than the TPR*-tree [17] (an improved variant of the TPR-tree) and the B^+-tree [3] based B^x-tree [6].

The rest of this paper is organized as follows. Section 2 surveys previous index techniques for moving objects and concurrency control for index trees. Section 3 and 4 describe the structure and algorithms of the Buddy*-tree. Experimental evaluation is described in Section 5. Finally, Section 6 concludes the paper.

2. RELATED WORK AND ANALYSIS

2.1 Index for Moving Objects

There is a long stream of research on the management and indexing of spatial and temporal data, which eventually led to the study of spatio-temporal data management. MOST [15] is one such early effort. By treating time as one dimension, moving objects in a d-dimension space can be indexed in $(d+1)$-dimensions. Thus, the predicted near future state of an object can be queried.

The *TPR-tree* (the Time Parameterized R-tree) [13] is a popular R-tree based index conceptually similar to MOST. Velocity vectors of objects or MBRs as well as the dynamic MBRs at current time are stored in the tree. At a non-leaf node, the velocity vector of the MBR is determined as the maximum value of velocities in each direction in the subtree. A good study of the performance of TPR-tree is in [17]. The *TPR*-tree* [17] was proposed to improve the TPR-tree by employing a new set of update algorithms. For insertions, TPR*-tree maintains a QP (priority queue) to record the candidates paths which have been inspected. By visiting the descendant nodes, TPR*-tree extends the paths in QP until a global optimal solution is chosen, while TPR-tree only chooses a local optimal path.

The B^x-tree [6] is a B$^+$-tree structure that indexesg moving objects after performing a transformation into single-dimensional space using a space filling curve. Objects are partitioned based on time, but indexed in the same space.

Indices based on hashing have been proposed to handle moving objects [16], [2]. Combinations have also been proposed. For instance, in [4], hashing on the grid cells is used to manage hot moving objects in memory, while the TPR-tree is used to manage cold moving objects on disk, as a way to provide efficient support for frequent updates.

2.2 Concurrency in the B-Tree and R-tree

The B-link tree [10] was proposed to provide efficient concurrent traversal and update of the B$^+$-tree. Every node keeps a right link pointing to the right sibling node in the same level. When a search process without lock-coupling goes down in the tree, it will learn of any splits racing with it by comparing keys. It is then able to visit the new split node along the right link chain before the new node is installed into the tree.

The R-link tree [9] employs a similar modification for the R-tree. The main difference between the R-tree and B$^+$-tree is that keys in R-tree are not ordered. Therefore, LSN (logical sequence number) is introduced to each node and kept in each entry of the internal nodes. Comparison of these LSNs is used to discover node splits. A right link chain is again used to locate newly split nodes.

2.3 MBR Expansion Due to Velocity

Let $x_i^l(0), x_i^u(0)$ be the lower bound and upper bound of some MBR respectively on dimension i at time 0, and $\vec{\boldsymbol{u}}_i^l, \vec{\boldsymbol{u}}_i^u$ be the minimum and maximum velocity of the points in the MBR on dimension i . After t time units, the volume of this MBR is $V = \prod_{i=1}^d (x_i^u(t) - x_i^l(t))$. Since $x_i^l(t) = x_i^l(0) + \vec{\boldsymbol{u}}_i^l \cdot t$ and $x_i^u(t) = x_i^u(0) + \vec{\boldsymbol{u}}_i^u \cdot t$, the volume of MBR can be rewritten as $V = \prod_{i=1}^d [(x_i^u(0) - x_i^l(0)) + (\vec{\boldsymbol{u}}_i^u - \vec{\boldsymbol{u}}_i^l) \cdot t]$. Therefore,

$$\frac{\partial V}{\partial t} = \sum_{i=1}^{d} \{(\vec{\boldsymbol{u}}_i^u - \vec{\boldsymbol{u}}_i^l) \cdot \prod_{i'=1, i' \neq i}^{d} [(x_i'^u - x_i'^l) + (\vec{\boldsymbol{u}}_i'^u - \vec{\boldsymbol{u}}_i'^l) \cdot t]\}$$

That is, $\frac{\partial V}{\partial t}$ is $O(t^{d-1})$.

The probability of any MBR being accessed by a random point search query, assuming uniform distributions, is proportional to the volume of the MBR. Therefore the expected number of MBRs accessed at any level of the index tree is proportional to the sum of their volumes. The rate of the increase on the expected number of MBRs to be accessed at some level l is $O(t^{d-1})$, where t is the elapsed time and d is the dimensionality. This shows that traditional indexing methods for moving object deteriorate greatly due to the overlap problem if there is no update for a long time.

3. OVERALL INDEXING STRUCTURE

3.1 Indexing Snapshots

In this paper, we adopt the basic assumption of maximum update interval of the moving objects. Under such an assumption, a moving object must update its movement at least once in every T_{ui} timestamps.

Thus, a series of snapshots indices are constructed on different reference timestamps. The reference timestamp of the ith index is on $T_{ui}(i - 0.5)$. There is a Buddy*-tree indexed at these reference timestamps for the moving objects. The detail of Buddy*-tree will be covered later in this paper. The lifespan of the ith index tree is between $T_{ui}(i-1)$ and $T_{ui}(i+1)$. Thus, there are two indexing trees maintained at the same time in our system. Figure 1 shows an example of the indexing trees.

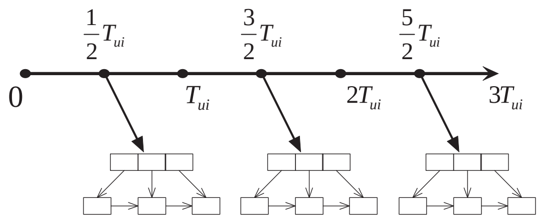


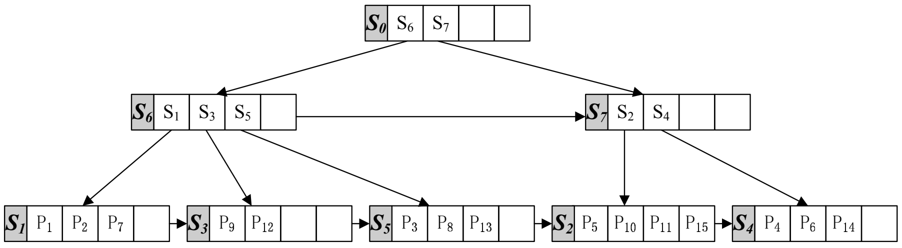
Figure 1: The indices on the snapshots

In our system, every object is indexed by one Buddy*-tree at a single timestamp. Assume an object updates its motion at timestamp t_u, it will be received and updated by the $(\lfloor t_u/T_{ui} \rfloor + 1)$th Buddy*-tree, according to the velocity and the position of the object at the reference time of the Buddy*-tree. If the object is previously indexed by the ith Buddy*-tree but the new update occurs on the $i+1$th tree, the object will be deleted from the ith tree as well as inserted into $i+1$th tree. For example, if $T_{ui} = 120$ and an object updates at $t_u = 110$, the update will be conducted on the first Buddy*-tree at timestamp 60. If it updates again at $t_u = 130$, the first tree will delete the object, while the second tree at timestamp 180 will insert the object into itself. It is guaranteed that the ith index tree must be empty at the end of its lifespan, since every object will update at least once between timestamp $T_{ui}i$ and $T_{ui}(i+1)$.

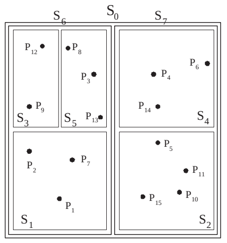
If the user issues a predictive range query q on timestamp t_q, our system will transform the queries to both the index trees currently maintained. Since every object is stored in at least one tree, the union of the range query result on these two trees must be a complete and valid result for the original query. The detail of the query processing on the Buddy*-tree will be covered later.

3.2 Buddy*-Tree

Given that we regard the moving objects as static points to index in each snapshot, and given the importance of fast update, we choose the Buddy-tree [14] as the basic structure of our proposed index. The index tree is constructed by cutting the space recursively into two subspaces of equal size with hyperplanes perpendicular to the axis of each dimension. Each subspace is recursively partitioned until the points in the subspace fit within a single page on



(a) The Buddy*-Tree



(b) A planar representation

Figure 2: An Example of the Structure of Buddy*-Tree

disk.

We make several alterations to this basic Buddy-tree structure to suit our needs. We call the new index structure Buddy*-tree. A traditional Buddy-tree creates tight bounding rectangles around the data points in each node. Since such tight MBRs are costly to update, we choose instead to use loose bounding. We call this a *Loose Bounding Space* (LBS) associated with the index tree node. We also store the maximum and minimum velocities in a node for all of the objects stored in the subtree of the node.

To support high degree concurrent operations on Buddy*-tree, we absorb the idea of right links among each level from B-link tree [10] and R-link tree [9]. Thus, at any given level all nodes are chained into a singly-linked list. In [9], the authors extends the R-Tree by assigning an additional parameter LSN as the timestamp to each node, which is used to detect the split and determine where to stop when moving right along the right link chain. However, this structural addition is not necessary in the Buddy*-tree since we are guaranteed not to have overlaps between nodes. Instead, we can simply detect the uninstalled node split by comparing the current LBS of the nodes with the old LBS. Figure 2(a) shows an example of the Buddy*-Tree in 2-dimensional space, with the corresponding data space illustrated in Figure 2(b). The first capital letter in a node denotes the LSB of it, followed by the entries with key LSB (expected LBS for the child node) or points.

3.3 Query Expansion on Buddy*-Tree

On every snapshot Buddy*-tree index, we use query expansion instead of MBR expansion for query answering. Our central idea is that movement of objects can be handled by expanding queries rather than actually perturbing objects in the index.

As in so many other moving object index structures, we use linear interpolation to estimate object position at times other than t_{ref}. The position of an object at time t can be calculated by the function $\boldsymbol{x}(t) = \boldsymbol{x}(t_{ref}) + \vec{\boldsymbol{v}} \times (t - t_{ref})$.

Based on this, we can suitably enlarge a query as follows: Suppose the query is q with query window $[qx_i^l, qx_i^u]$ ($i = 0, 1, ...d - 1$,where d is dimension of the space), and the query time is t_q, the enlarged query window $[eqx_i^l, eqx_i^u]$ is obtained as:

$$eqx_i^l = \begin{cases} qx_i^l + \vec{\boldsymbol{u}}_i^l \cdot (t_{ref} - t_q) & if t_q < t_{ref} \\ qx_i^l + (-\vec{\boldsymbol{u}}_i^u) \cdot (t_q - t_{ref}) & otherwise \end{cases}$$

$$eqx_i^u = \begin{cases} qx_i^u + \vec{\boldsymbol{u}}_i^u \cdot (t_{ref} - t_q) & if t_q < t_{ref} \\ qx_i^u + (-\vec{\boldsymbol{u}}_i^l) \cdot (t_q - t_{ref}) & otherwise \end{cases}$$

where $\boldsymbol{u}_i^l$ and $\boldsymbol{u}_i^u$ are the minimum and maximum velocities respectively of objects inside the query window in dimension i. Note that we would ideally have liked to enlarge the query by precisely the velocities of the objects included in the query. Although we do not have idea about which objects are in the query result, the maximum and minimum velocity stored in the nodes is enough to help us find the correct result. The following theorem is straightforward and used in next section.

THEOREM 1. *Given a query q and a MBR node N, the enlarged query q' must overlap N if N contains at least one object in the result of q.*

4. BUDDY*-TREE OPERATIONS

In this section, we provide the detailed algorithm on the operations of Buddy*-tree, including querying, insertion and deletion. We also discuss how to achieve high concurrency on Buddy*-tree.

There are three kinds of locks on the nodes of Buddy*-tree, read lock, write lock and mark lock. If a node is read locked, this node can be read by other threads but can not be written. If a node is write locked, it can not be read or written by any other thread. If the node is mark locked, all read and write operations, except merge, can be run on it. In the following, we use r_lock (r_unlock), w_lock (w_unlock) and m_lock (m_unlock) to denote the locking (unlocking) operations for read lock, write lock and mark lock respectively.

4.1 Querying

Algorithm 1 Range_Search(r, N, l)

```
/* Input: $r$ is the query window. $N$ and $l$ are the pointer of the node to be
examined and its LBS obtained from its parent node, respectively*/
1: r_lock($N$)
2: if $N$ is mark locked by this thread then
3:    m_unlock($N$)
4: for each entry $e$ in $N$ that $e.LBS$ overlaps $R$, obtained by enlarging $r$
   according to the time difference and extremal velocities in $e.node$ do
5:    if $N$ is not a leaf node then
6:       Insert ($e.node$, $e.LBS$) into $tobeVisited$
7:       m_lock($e.node$)
8:    else
9:       output qualified points in $e$
10: r_unlock($N$)
11: while $tobeVisited$ is not empty do
12:    ($N'$, $l'$) = GetFirst($tobeVisit$)
13:    Range_Search($r$, $N'$, $l'$)
14: if $N.LBS$ is not equal to $l$ then
15:    traverse the right link chain starting at $N$ to first node whose LBS is
       not contained in $l$
16:    for each node $M$ along the chain except the last one do
17:       r_lock($M$)
18:       $l'$ := $M.LBS$
19:       r_unlock($M$)
20:       Range_Search($r$, $M$, $l$)
```

In Algo 1, we provide the detail of the recursive range search over Buddy*-tree. The searching process is similar to other tree-indexing structure. For a non-leaf node in Buddy*-tree, the algorithm retrieves all the children with overlap with the expanded query, and put all these children into the list for nodes to be visited later (line 5 to line 13). Then, the algorithm further visits and search the new nodes created by other threads on the right link chain of the current node (line 14 to line 20).

By Theorem 1, the querying algorithm can not miss any node containing at least one point in the query result. Therefore, it can always output the correct result in all cases.

The read lock is exerted on the node when the range search algorithm tries to retrieves the children node. Such a lock is used also during the process of searching uninstalled nodes on the right chain to get the correct LBS.

4.2 Insertion

To insert a point into Buddy*-tree, our system first computes the location of the point at the reference time of the tree. Then, the simple location searching operation is invoked to locate the leaf node in Buddy*-tree whose LBS covers the point. In Algo 2, we present the recursive implementation of how to insert an entry into a Buddy*-tree node. Given the node to insert the point, the algorithm first finds the correct node if uninstalled split nodes exist (line 1 to line 4). If the node to insert still has room for a new entry, the algorithm directly inserts it into it (line 5 to line 7). Otherwise, some split operations are invoked, which is followed by recursive insertion operation on the parent of the node (line 9 to line 18).

Algorithm 2 Insert_Entry(s, N)

```
/* Input: $s$ is the entry containing a point or a branch to install into node $N$.
Nodes $N$ as input is write-locked and it is unlocked after the procedure, */
 1: while $N.LBS$ doesn't cover $s.LBS$ do
 2:     $N'$ := $N$'s right neighbor on the chain.
 3:     w_lock($N'$) and w_unlock($N$)
 4:     $N$ := $N'$
 5: if there is an empty entry $e$ in $N$ then
 6:     put $s$ in $e$
 7:     w_unlock($N$)
 8: else
 9:     $newN$ = Split_Node($N$) /* Split $N$ into two nodes $N$ and $newN$ */
10:     Choose one from $N$ and $newN$ to insert $s$
11:     if $N$ is not root then
12:         $P$ := $N$'s parent node
13:         w_lock($P$) and w_unlock($N$)
14:         $e$ := the entry containing $newN$
15:         Insert_Entry($e$, $P$)
16:     else
17:         Construct a new root and insert $N$ and $newN$ into it
18:         w_unlock($N$)
```

The write lock is exerted on a node in two cases. The first case happens when the algorithm is trying to insert an entry to the node. The second case happens when the algorithm finds an uninstalled split, it moves the cursor as well as the write lock to the right nodes on the chain.

4.3 Deletion

The deletion operation is similar to insertion operation. In the first step, the leaf node containing s is first located, followed by Algo 3 to remove the entry and merging its parent if necessary.

In Algo 3, we present the detailed process of deleting an entry from the tree. The algorithm first locates the correct node by traversing on the right chain (line 1 to line 4). The entry containing the point is directly removed if it is found in the correct node (line 5 to line 8). Once the resulting node has too few entries, merging operations is invoked if the right neighbor on the right chain is its

Algorithm 3 Delete_Entry(s, N)

```
/* Input: $s$ is the entry containing a point or a branch to install into node $N$.
Nodes $N$ as input is write-locked and it is unlocked after the procedure, */
 1: while $N.LBS$ doesn't cover $s.LBS$ do
 2:     $N'$ := $N$'s right neighbor on the chain.
 3:     w_lock($N'$) and w_unlock($N$)
 4:     $N$ := $N'$
 5: if find the entry $s$ in $N$ then
 6:     delete $s$ from $N$
 7: else
 8:     w_unlock($N$) and report error
 9: if there are too few entries in $N$ then
10:     $M$ := $N$'s right neighbor on the right chain
11:     w_lock($M$) /* delayed if $M$ is mark locked */
12:     if $M$ and $N$ are buddies and the total number of entries in $M$ and
        $N$ below maximum entry bound then
13:         $P$ := $N$'s parent node
14:         w_lock($P$)
15:         Merge $M$ into $N$
16:         $e$ :=the entry containing $M$
17:         Del_Entry($e$, $P$)
18:     else
19:         w_unlock($M$)
20: w_unlock($N$)
```

buddy in the Buddy*-tree. The merging process is finished after the parent of the two nodes removes one of the buddies (line 9 to line 19). We note that the merge operation can be delayed due to the mark lock exerted by querying operation.

Write lock is exerted on at most three nodes in the algorithm, including the node N containing s, N's right neighbor M and their parent P.

4.4 Deadlock Analysis

There can be no deadlock between any two threads in the system. We analyze the situations based on the operations of the threads as follows.

Two threads of the same type of operations must be safe from deadlock, since the threads must lock the nodes on the same direction. For one querying thread and one insertion thread, there can be no deadline, since the querying node can read lock at most one node at the same time, while the insertion node can wait for the release of it. For one insertion thread and one deletion thread, they are also deadlock free, since both threads work from bottom to top on the tree. For one querying thread and one deletion thread, on the first hand, the read lock from querying thread can not have deadlock with the write lock from deletion thread. On another hand, the mark lock is released after the querying thread exerting the read lock, so it can not block deletion thread either.

5. EXPERIMENTAL EVALUATION

We implemented the Buddy*-tree, and compared its performance to that of the TPR*-tree and B^x-tree. All of these structures were implemented in C. All experiments were conducted on a single CPU 3G PentiumIV Personal Computer with 1 G bytes of memory.

We ran two sets of experiments, one with a single thread of activity, and another with multiple concurrent threads. In both sets of experiments we use synthetic uniform datasets. The position of each object in the data set is chosen randomly in a 1000×1000 space. Each object moves in a randomly chosen direction with a randomly chosen speed ranging from 0 to 3. We constructed the index at time 0. The parameters used in the experiments are summarized in Table 1, and the default values are highlighted in bold.

Parameter	Setting
Page size	4K
Max update interval	60,**120**,180,240
Max predictive interval	120
Query window size	**10**,20,...,100
Number of queries	200
Dataset size	100K,...,**500K**,...1M
Number of threads	2,4,8,...,**64**,128,256
Number of operations per thread	200

Table 1: Parameters and Settings

5.1 Storage Requirement

In a Buddy*-tree internal entry, the space partition is kept for the child node (at least 8 Bytes for 2-dimension space). As for B^x-tree, each entry contains a 64bit key (8 Bytes). However, a TPR*-tree internal node stores MBRs and VBRs for each child entry (24 Bytes for 2-dimensions). The storage requirement of the indices is shown in Figure 3. As anticipated, TPR*-tree requires more than twice storage space of the others, which are comparable.

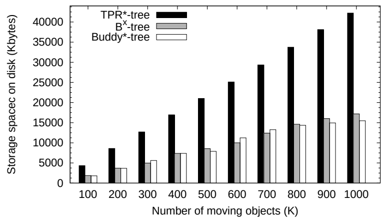
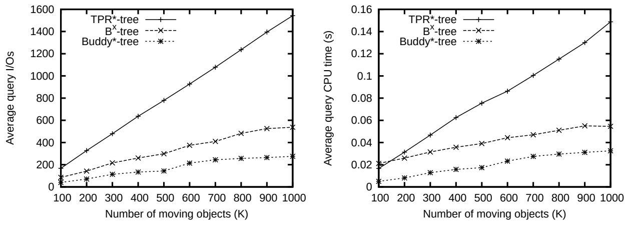


Figure 3: Storage Requirement

5.2 Single Thread Experiments

5.2.1 *Effect of Dataset Size*

First, we study the range query performance with different sizes of dataset. 200 window queries with size 10 are issued after the index running for 120 time units (an entire maximum update interval). The predictive intervals of the queries are randomly chosen in the range from 0 to 120. Figure 4 shows the average cost of I/O operation and CPU time per query for the three inspected indices.



(a) I/O cost (b) CPU time

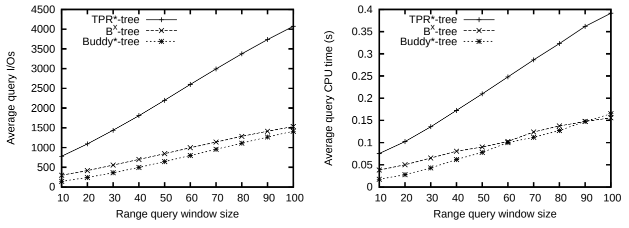
Figure 4: Effect of Dataset Size on Range Query Performance

As expected, the results show that the window query costs of all the indices increase with the number of objects. However, The increasing speed of TPR*-tree is much higher than that of the others. When there are 1M objects in the dataset, the cost of the TPR*-tree is nearly 3 times over that of B^x-tree and more than 5 times over that of Buddy*-tree. This is due to the fact that, the performance of TPR*-tree highly depends on the ratio of overlap, while that of B^x-tree and Buddy*-tree is related only to the result sizes of the queries.

5.2.2 *Effect of Query Size*

We next investigate the performance of the indices with respect to query size.

In the experiments we vary the query window size from 10 to 100 on a dataset of size 500K. The predictive intervals of queries are set as same with last experiments. As shown in Figure 5, query costs increase with the query window size. This behavior is straightforward, since a larger window contains more objects and accordingly, more index nodes will be accessed. The TPR*-tree degenerates considerably over the other indices. This behavior is attributed to the overlap problem of TPR*-tree. The costs of B^x-tree and Buddy*-tree increase at the same rate.

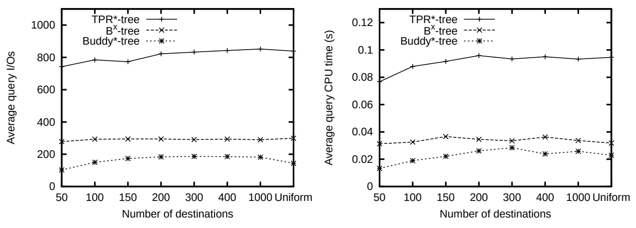


(a) I/O cost (b) CPU time

Figure 5: Effect of Query Window Size on Window Query Performance

5.2.3 *Effect of Data Distribution*

This experiment uses the network dataset to study the effect of data distribution on the indexes. The dataset is generated by an existing data generator, where objects move in a road network of two-way routes that connect a given number of uniformly distributed destinations [13]. The dataset contains 500K objects, that are placed at random positions on routes and are assigned at random to one of three groups of objects with maximum speeds of 0.75, 1.5, and 3. Objects accelerate as they leave a destination, and they decelerate as they approach a destination. Whenever an object reaches its destination, a new destination is assigned to it at random.



(a) I/O cost (b) CPU time

Figure 6: Effect of Data Distribution on Range Query Performance

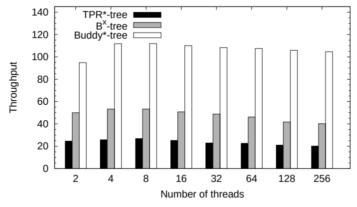
Figure 6 summarizes the average range query costs of the three indexes when the number of destinations in the simulated network of routes is varied. Decreasing the number of destinations adds skew to the distribution of the object positions and their velocity vectors. As is shown, increased skew leads to a decrease in the range query cost in the TPR*-tree, since when there are more objects with similar velocities, they are easier to be bounded into rectangles. The performance of the B^x-tree is not affected by the data skew because objects are stored using space-filling curves, which is not sensitive to density. Observe that the range query cost of the Buddy*-tree firstly increases with the number of destinations and after the point that the number of destination is 300, the cost descends.

5.3 Concurrent Operations

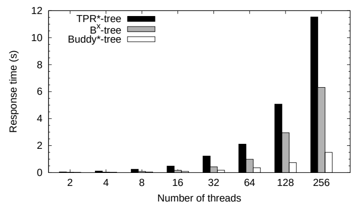
We implemented B-link for B^x-tree, and simply locked the whole TPR*-tree [1]for concurrency control. In the following experiments, each thread issues 200 operations, and the workload of each thread contains the same number of queries and updates.

5.3.1 Effect of Thread Number

First, we investigate the effect of the number of threads. Figure 7 shows the throughput and response time for the indices by varying the thread number from 2 to 256.



(a) Throughput



(b) Response time

Figure 7: Effect of Threads on Concurrent Operations

All the indices reach the highest throughput at around 8 threads and thereafter show deteriorating performance as the number of threads is increased. Measuring the decline as we go from 8 to 256 threads, we find this decrease to be only 6.5% for Buddy*-tree, but 24.5% and 24.6% for B^x-tree and TPR*-tree respectively. Since Buddy*-tree has been designed for high concurrency, its superior performance with multiple threads validates our design. The decline in performance of TPR*-tree is also to be expected. The surprise is the decline in performance of the B^x-tree in spite of the use of B-link chain for high concurrency. The main reason for this is that a lot of "jumps" in the B^x-tree for range query increase the number of accesses of and locks on internal nodes.

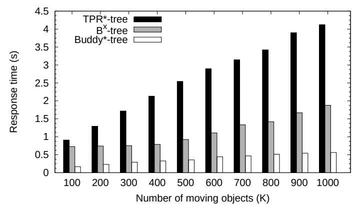
5.3.2 Effect of Dataset Size

As shown in Figure 8, the performance of all indices reduces with the increasing number of moving objects. This is straightforward, since the larger the dataset is, the more nodes an index contains and the more I/O operations a query or update brings. However the Buddy*-tree outperforms the other indices for both throughput and response time.

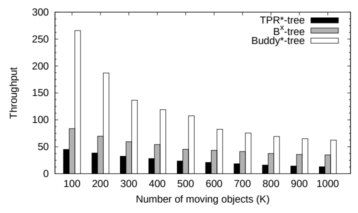
6. CONCLUSION

In this paper, we proposed a space partitioned based index structure Buddy*-tree, a generalization of Buddy-tree, for indexing mobile objects. An adaptive query expansion technique is used for predictive range query over object motion, while only static snapshots indexed, with a right link structure to achieve higher concurrency.

[1]Since TPR*-tree employs different update algorithms from TPR-tree (e.g. remove and reinsert a set of entries in split algorithm), it can not grantee RR even we implement R-link for it.



(a) Throughput



(b) Response time

Figure 8: Effect of Data Size on Concurrent Operations

7. REFERENCES

[1] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. In *the Proceedings of ACM SIGMOD*, 1990.

[2] H. D. Chon, D. Agrawal, and A. E. Abbadi. Using space-time grid for efficient management of moving objects. In *the Second ACM international workshop on Data engineering for wireless and mobile access*, 2001.

[3] D. Comer. The ubiquitous b-tree. *ACM Computing Surveys*, 11(2):121–137, 1979.

[4] B. Cui, D. Lin, and K. L. Tan. Towards optimal utilization of main memory for moving object indexing. In *Proceedings of DASFAA*, 2005.

[5] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *the Proceedings of ACM SIGMOD*, 1984.

[6] C. S. Jensen, D. Lin, and B. C. Ooi. Query and update efficient b+-tree based indexing of moving objects. In *Proceedings of VLDB*, 2004.

[7] G. Kollios, D. Gunopulos, and V. J. Tsotras. Nearest neighbor queries in a mobile environment. In *Proceedings of the International Workshop on Spatio-Temporal Database Management*, 1999.

[8] G. Kollios, D. Gunopulos, and V. J. Tsotras. On indexing mobile objects. pages 261–272, 1999.

[9] M. Kornacker and D. Banks. High-concurrency locking in r-trees. In *Proceedings of VLDB*, pages 134–145, 1995.

[10] P. Lehman and S. Yao. Efficient locking for concurrent operations on b-trees. *ACMTODS*, 6(4), Dec. 1981.

[11] J. Nievergelt, H. Hinterberger, and K. C. Sevcik. The Grid File: An adaptable, symmetric multikey file structure. *ACM TODS*, 9(1):38–71, 1984.

[12] B. C. Ooi, K. L. Tan, and C. Yu. Frequent update and efficient retrieval: an oxymoron on moving object indexes? In *Proceedings of International Web GIS Workshop*, 2002.

[13] S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez. Indexing the positions of continuously moving objects. In *the Proceedings of ACM SIGMOD*, pages 331–342, 2000.

[14] B. Seeger and H. P. Krieger. The buddy-tree: An efficient and robust access method for spatial data base systems. In *Proceedings of VLDB*, pages 590–601, Aug. 1990.

[15] A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and querying moving objects. In *the Proceedings of ICDE*, pages 422–432, 1997.

[16] Z. Song and N. Roussopoulos. Hashing moving objects. In *the Proceedings of the 2nd International Conference on Mobile Data Management*, 2001.

[17] Y. Tao, D. Papadias, and J. Sun. The TPR*-tree: An optimized spatio-temporal access method for predictive queries. In *the Proceedings of VLDB*, 2003.

An Online Bibliography on Schema Evolution

Erhard Rahm
University of Leipzig
rahm@informatik.uni-leipzig.de

Philip A. Bernstein
Microsoft Corporation
philbe@microsoft.com

Abstract

We briefly motivate and present a new online bibliography on schema evolution, an area which has recently gained much interest in both research and practice.

1 Introduction

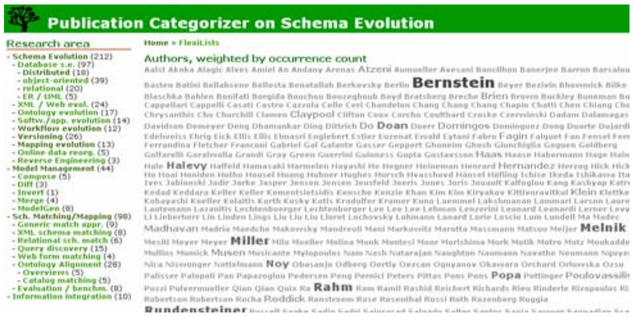
Schema evolution is the ability to change deployed schemas, i.e., metadata structures formally describing complex artifacts such as databases, messages, application programs or workflows. Typical schemas thus include relational or object-oriented (OO) database schemas, conceptual ER or UML models, ontologies, XML schemas, software interfaces and workflow specifications. Obviously, the need for schema evolution occurs very often in order to deal with new or changed requirements, to correct deficiencies in the current schemas or to migrate to a new platform.

Effective support for schema evolution is challenging since schema changes may have to be propagated, correctly and efficiently, to instance data, views, applications and other dependent system components. Ideally, dealing with these changes should require little manual work and system unavailability. For instance, changes to a database schema S should be propagated to instances data and views defined on S with minimal human intervention.

Schema evolution has been an active research area for a long time. However, the need for powerful schema evolution has been increasing and many papers have appeared recently. Moreover, commercial database systems such as IBM DB2, Oracle and Microsoft SQL Server have started to support online schema evolution capabilities. Thus, although several previous bibliographies and surveys exist [13, 14], there is value in producing an up-to-date bibliography.

One reason for increased interest is that widespread use of XML and web services has led to new schema types and usage scenarios of schemas for which schema evolution must be supported. For example, data integration architectures, such as enterprise information integration (EII) and enterprise application integration (EAI), are now common and must be able to deal with schema changes for data sources, global schemas and ontologies. The growing importance of such problems has prompted recent work on generic metadata management, e.g., schema matching and model management. Such approaches can help automate schema evolution tasks, e.g., generation and adaptation of mappings between schemas. The goal of our online bibliography is to provide a comprehensive and up-to-date collection of publications on schema evolution. We are not limiting ourselves to database schema evolution but also consider related fields such as ontology evolution, software evolution and workflow evolution. We have found these evolution problems are often similar so that proposed solutions may be transferable to different fields. For example, ontology evolution has similarities to previously investigated evolution problems in object-oriented database systems.

Our bibliography on schema evolution is accessible on the web under **http://se-pubs.dbs.uni-leipzig.de**. We use our content categorization tool Caravela [1] to categorize publications along multiple hierarchical dimensions. Bibliographic entries typically contain the abstract, full-text link and current number of citations from Google Scholar. The system provides many ways to search and browse the categories and papers. Publication entries can be added, corrected and categorized collaboratively (wiki-like) by many users. Automated data import from files or web sites like Google scholar is also supported for authorized users. Fig. 1 shows the current start page of the bibliography indicating the categories (on the left) and author names sized according to the number of available papers.



Publication Categorizer on Schema Evolution

Research area
- Schema Evolution (212)
  - Database s.e. (97)
    - Distributed (18)
    - object-oriented (39)
    - relational (20)
    - ER / UML (5)
  - XML / Web evol. (24)
  - Ontology evolution (17)
  - Softw./app. evolution (14)
  - Workflow evolution (12)
  - Versioning (26)
  - Mapping evolution (13)
  - Online data reorg. (5)
  - Reverse Engineering (3)
- Model Management (44)
  - Compose (5)
  - Diff (3)
  - Invert (1)
  - Merge (4)
  - ModelGen (8)
- Sch. Matching/Mapping (98)
  - Generic match appr. (9)
  - XML schema matching (8)
  - Relational sch. match (6)
  - Query discovery (15)
  - Web form matching (4)
  - Ontology Alignment (28)
    - Overviews (5)
    - Catalog matching (5)
  - Evaluation / benchm. (8)
- Information integration (10)

Home » FlexiLists

Authors, weighted by occurrence count

Aalst Akoka Alagic Alves Amiel An Andany Arenas Atzeni Aumueller Aversani Bancilhon Banerjee Barron Barsalou
Basten Batini Bellahsene Bellosta Benatallah Berkovsky Berlin **Bernstein** Beyer Bezivin Bhowmick Bilke
Blaschka Bohlen Bonifati Borgida Bouchou Bouzeghoub Boyd Bratsberg Breche Brien Brown Buckley Buneman Bu
Cappellari Cappelli Casati Castro Cazzola Celle Ceri Chandelon Chang Chang Chang Chapin Chatti Chen Chiang Ch
Chrysanthis Chu Churchill Clamen Claypool Clifton Coox Corcho Coulthard Craske Czerwinski Dadam Dalamagas
Davidson Demeyer Deng Dhamankar Ding Dittrich Do **Doan** Duerr Domingos Dominguez Dong Duarte Dujardi
Edelweiss Ehrig Eick Ellis Ellis Elmasri Englebert Estier Euzenat Ewald Eytani Fabro Fagin Falquet Fan Fensel Fer
Ferrandina Fletcher Franconi Gabriel Gal Galante Gasser Geppert Ghoneim Ghosh Giunchiglia Gegoen Goldberg
Golfarelli Goralwalla Grandi Gray Green Guerrini Guinness Gupta Gustavsson Haas Haase Habermann Hage Hal
Hale **Halevy** Halfeld Hamasaki Harmelen Hayashi He Hegner Heineman Henrard Hernandez Herzog Hick Hick
Hu Hoai Honiden Hotho Housel Huang Hubner Hughes Hursch Hvasshovd Hänsel Höfling Ichise Ikeda Ishikawa Ita
Ives Jablonski Jadir Jarke Jasper Jensen Jensen Jeusfeld Jeeris Jones Joris Jouault Kalfoglou Kang Kashyap Katr
Kedad Keddara Keller Keller Komentsietsidis Kensche Kenzie Khan Kim Kim Kiryakov Kittivoravitkul Klein Klettke
Kobayashi Koeller Kolaitis Korth Kosky Kotis Kradolfer Kramer Kuno Laemmel Lakshmanan Lammari Larson Laur
Lautemann Lazanitis Lechtenboerger Lechtenburger Lee Lee Lee Lehman Lenzerini Leonard Leonardi Lerner Levy
Li Lieberherr Lin Linden Lings Liu Liu Liu Lloret Lochovsky Lohmann Lonard Lorie Loscio Lum Lundell Ma Madec
Madhavan Madria Maedche Makowsky Mandreoli Mani Markowitz Marotta Massmann Matsuo Meijer **Melnik**
Mesiti Meyer Meyer **Miller** Milo Moeller Molina Monk Montesi Moor Morishima Mork Motik Motro Mutz Moukadd
Mullins Munick Musen Musicante Mylopoulos Nam Nash Natarajan Naughton Naumann Navathe Neumann Nguye
Nica Niswonger Nottelmann **Noy** Obasanjo Odberg Oertly Oezcan Ognyanov Okawara Orchard Orlowska Ozsu
Palisser Palopoli Pan Paparoglou Pedersen Peng Pernici Peters Pittas Pons Pons **Popa** Pottinger Poulovassili
Pozzi Pulvermueller Qian Qian Quix Ra **Rahm** Ram Ramil Rashid Reichert Richards Rieu Rinderle Rizopoulos Ri
Robertson Robertson Rocha Roddick Runstroem Ruse Rosenthal Rossi Roth Rozenberg Ruggia
Rundensteiner [illegible]

As of October 2006 more than 300 papers on schema evolution and related fields are categorized. We broadly assign papers within the following categories:

- Database schema evolution
- XML schema evolution
- Ontology evolution
- Software evolution and
- Workflow evolution.

Furthermore we have separate categories for popular solution approaches, such as

- Schema versioning and
- Model and mapping management.

Each category is usually divided into several subcategories. Using our categorization tool these categories can easily be extended or refined as needed, i.e., we support evolution of the categorization schema. In the following we briefly list some specific aspects of the research categories covered by the bibliography.

2 Research categories

2.1 Database schema evolution

Some papers characterize types of schema changes for different data models, in particular relational, object-oriented (OO) and XML databases. These changes can be propagated to instances of the schema immediately or lazily. They may also be propagated to dependent views, something that today's commercial database systems do automatically for simpler schema changes (e.g., 1 table).

There are many papers on schema evolution for OO database systems [2,11], since evolution is intrinsic to the design processes they support. By contrast, there are still relatively few papers on schema evolution in distributed systems—possibly a good opportunity for future research.

2.2 XML schema evolution

The semi-structured nature of XML offers more flexibility in coping with schema changes and lower cost, due to such features as optionality of schema parts and multiple schemas per database [4].

2.3 Ontology evolution

Ontologies exhibit the same evolution problems as database schemas, but have some different constructs, such as controlled vocabularies, taxonomies, and rule-based knowledge representation, and hence have some different types of changes. Often, an ontology contains both schema-like conceptual metadata plus its instances; changes to metadata and instances need to be considered together. A domain ontology may be used in many applications, resulting in dependencies between distributed systems. So far, most papers focus on ontology matching and versioning aspects of evolution [6, 7].

2.4 Software evolution

The generation of a new software version shares many of the problems of schema evolution. Instead of schemas, we have program interfaces or class hierarchies. Instead of mappings or views, we have usage relationships and dependencies between program modules. Research papers classify different software evolution and maintenance scenarios [8]. Many papers focus especially on object-oriented software development [9]. Some describe change support tools.

2.5 Workflow evolution

Workflows are long-running activities. Instead of schemas and databases, we have workflow specifications and executing workflow instances. So changing a workflow specification (e.g., change/add/drop an activity) requires different actions than changing a database schema [5].

2.6 Version management

One major approach to schema evolution is the use of user-controlled, explicit versions [7, 14]. For example, the need to propagate changes is reduced by preserving older versions of schemas. Although versioning is rarely used for database schema evolution, it is a very common approach to software evolution and will likely be important for XML, web service, and ontology evolution.

2.7 Model and mapping management

High-level operators on schemas and mappings are useful for generating views and other mappings and adapting them after schema changes.

- There is a big literature on schema matching [12], which can help determine what has changed. Schema evolution is a simple case for schema matching since most of the schema remains unchanged.
- Given a result from schema matching, there are query discovery techniques [10] to generate an executable (instance-level) mapping between the old and evolved schema.
- Given a mapping from an evolved schema to the old schema and an existing view over the old schema, mapping composition can be used to produce an updated view [15].
- Scripts of match, compose and other operators have been published for a variety of complex schema evolution scenarios [3].

3 References

1. Aumüller, D., Rahm, E.: Caravela: Semantic Content Management with Automatic Categorization. Univ. of Leipzig, 2006
2. Banerjee, J.; Kim, W.; Kim, H.; Korth, H. F. Semantics and Implementation of Schema Evolution in Object-Oriented Databases. *Proc. SIGMOD 1987*
3. Bernstein, P.A.: Applying Model Management to Classical Meta Data Problems. *Proc. CIDR 2003*
4. Beyer, K.; Oezcan, F.; Saiprasad, S.; Van der Linden, B.: DB2/XML: Designing for Evolution. *Proc. SIGMOD 2005.*
5. Casati, I.; Ceri, S.; Pernici, B.; Pozzi, G. Workflow Evolution. *Proc. Int. Conf. on Conceptual Modeling (ER)*, 1996
6. Doan, A.; Madhavan, J.; Domingos, P.; Halevy, A.: Learning to Map between Ontologies on the Semantic Web. Proc. *WWW2002*
7. Klein, M.; Fensel, D.: Ontology Versioning on the Semantic Web. *Proc. Int. Semantic Web Working Symposium*, 2001
8. Lehman, M.M; Ramil, J.F.: Software Evolution - Background, Theory, Practice. *Inf. Process. Lett.* 88(1-2), 2003
9. Lieberherr, K.J., Xiao, C.: Object-Oriented Software Evolution. *IEEE Trans. Software Eng.* 19(4), 1993
10. Miller, R.; Haas, L.; Hernandez, M.: Schema Mapping as Query Discovery. *Proc. 26th VLDB*, 2000
11. Ra, Y; Rundensteiner, E.: A Transparent Schema-Evolution System Based on Object-Oriented View Technology. *IEEE Trans. Knowledge and Data Eng 9(4)*, 1997
12. Rahm, E.; Bernstein, P. A.: A Survey of Approaches to Automatic Schema Matching. *VLDB Journal*, 2001
13. Roddick, J.F.: Schema Evolution in Database Systems: An Annotated Bibliography. *SIGMOD Record 21(4)*, 1992
14. Roddick, J.F.: Survey of Schema Versioning Issues for Database Systems. *Information and Software Technology*, 37(7), 1995
15. Yu, C.; Popa, L.: Semantic Adaptation of Schema Mappings when Schemas Evolve. *Proc. VLDB 2005*

The ADO.NET Entity Framework: Making the Conceptual Level Real

José A. Blakeley, David Campbell, S. Muralidhar, Anil Nori
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399, USA
{joseb, davidc, smurali, anilnori}@microsoft.com

ABSTRACT

This paper describes the ADO.NET Entity Framework, a platform for programming against data that raises the level of abstraction from the logical (relational) level to the conceptual (entity) level, and thereby significantly reduces the impedance mismatch for applications and data services such as reporting, analysis, and replication. The conceptual data model is made real by a runtime that implements an extended relational model (the Entity Data Model aka the EDM), that embraces entities and relationships as first class concepts; a query language for the EDM; a comprehensive mapping engine that translates from the conceptual to the logical (relational) level, and a set of model-driven tools that help create entity-object, object-xml, and entity-xml transformers.

1. INTRODUCTION

Modern applications require data management services in all tiers. They need to handle increasingly richer forms of data which includes not only structured business data (customers, orders) but also XML, email, calendar, files, and documents. These applications need to integrate data residing in multiple data sources and enable end-to-end business insight by collecting, cleaning, storing, and preparing business data in forms suitable for an agile decision making process. Developers of these applications need data access, programming and development tools to increase their productivity.

This paper describes the ADO.NET Entity Framework, a platform for programming against data that significantly reduces the impedance mismatch for applications and data services such as reporting, analysis, and replication. We argue that modern applications and data services need to target a higher-level conceptual model based on entities and relationships rather than the relational model and that such a conceptual model needs to be implemented concretely in a data platform. The Entity Framework makes the conceptual data model concrete by a runtime that implements an extended relational model – the Entity Data Model, or the EDM - that embraces entities and relationships as first class concepts, a query language for the EDM, a comprehensive mapping engine that translates from the conceptual to the logical (relational) level, and a set of model-driven tools that help create entity-object, object-xml, and entity-xml transformers. The Entity Framework is part of a broader Microsoft Data Access vision supporting a family of *products and services so customers derive value from all data, birth through archival.*

Section 2 describes the physical, logical, conceptual and programming levels as well as other terms used throughout the paper. Section 3 describes the evolution of applications and data services and motivates the need for making the conceptual level central to application and data services design. Section 4 introduces the Entity Data Model and the concrete manifestation of this model in the Entity Framework. Section 5 presents a summary and conclusions.

2. DATABASE MODELING LAYERS

Today's dominant information modeling methodology for producing database designs factors an information model into four main levels: Physical, Logical (Relational), Conceptual, and Programming/Presentation.

The ***physical*** model describes how data is *represented* in physical resources such as memory, wire or disk. The vocabulary of concepts discussed at this layer include record formats, file partitions and groups, heaps, and indexes. The physical model is typically invisible to the application - applications usually target the logical or relational data model described in the next section. Changes to the physical model should not impact application logic, but may impact application performance.

A ***logical*** data model is a complete and precise information model of the target domain. The relational model is the representation of choice for most logical data models. The concepts discussed at the logical level include tables, rows, and primary key-foreign key constraints, and normalization. While normalization helps to satisfy important application requirements such as data consistency and increased concurrency with respect to updates and OLTP performance, it also introduces significant challenges for applications. (Normalized) Data at the logical level is too fragmented and application logic needs to aggregate rows from multiple tables into higher level entities that more closely resemble the artifacts of the application domain. The conceptual level

This paper is a revised version of one that appeared in the Proceedings of the 25th International Conference on Conceptual Modeling, Tucson, AZ, USA, November 6-9, 2006, available from Springer at: http://www.springer.com/east/ home/generic/search/results?SGWID=5-40109-22-173696005-0. The authors thank Springer for their permission to publish this revised paper.

introduced in the next section is designed to overcome these challenges.

The ***conceptual*** model captures the core information entities from the problem domain and their relationships. A well-known conceptual model is the Entity-Relationship Model introduced by Peter Chen in 1976 [1]. UML is a more recent example of a conceptual model [2].

Most significant applications involve a conceptual design phase early in the application development lifecycle. Unfortunately, however, the conceptual data model is captured inside a database design tool that has little or no connection with the code and the relational schema used to implement the application. The database design diagrams created in the early phases of the application life cycle usually stay "pinned to a wall" growing increasingly disjoint from the reality of the application implementation with time. However, a conceptual data model can be as real, precise, and focused on the concrete "concepts" of the application domain as a logical relational model. A goal of the Microsoft Data Access vision is to make the conceptual data model (embodied by the Entity Data Model, described in Section 4.2) a concrete feature of the data platform.

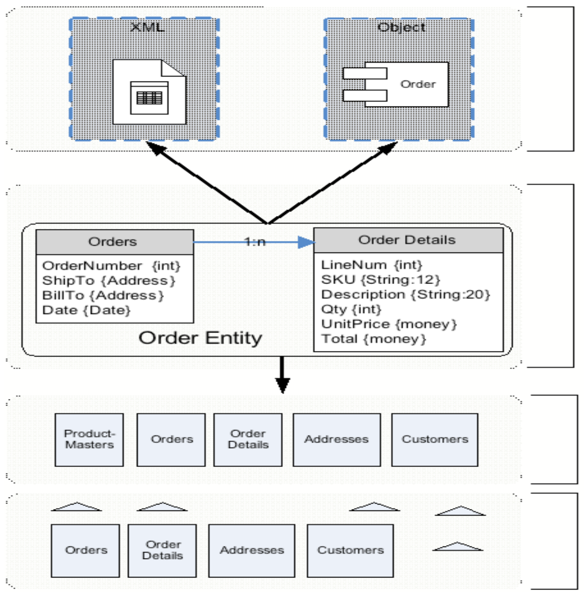


Figure 1: Physical, logical, conceptual and multiple programming and presentation views of an Order.

The ***programming/presentation*** model describes how the entities and relationships of the conceptual model need to be manifested (presented) in different forms based on the task at hand. Some entities need to be transformed into programming language objects to implement application business logic; others need to be transformed into XML streams for web service invocations; still others need to be transformed into in-memory structures such as lists or dictionaries for the purposes of user-interface data binding. Naturally, there is no universal programming model or presentation form; thus applications need flexible mechanisms to transform entities into the various presentation forms.

Most developers, and most of the modern data services want to reason about high-level concepts such as an "Order" (See **Error! Reference source not found.**), not about the several tables that an order may be normalized over in a relational database schema. They want to query, secure, program, report on the order. An order may manifest itself at the presentation/programming level as a class instance in Visual Basic or C# encapsulating the state and logic associated with the order, or as an XML stream for communicating with a web service. We believe there is no "one proper presentation model"; and that the real value is in making the conceptual level real and then being able to use that model as the basis for flexible mappings to and from various presentation models and other higher level services.

3. APPLICATION AND DATA SERVICES EVOLUTION

This section describes the platform shift that motivates the need for a higher level data model and data platform. We will look at this through two perspectives: application evolution and SQL Server's evolution as a product. A key point we make in this section is that the need for rich data model is motivated not just for developing application logic but also for supporting building higher-level data services such as reporting and replication.

3.1 Application Evolution

Data-based applications 10-20 years ago were typically structured as data monoliths; closed systems with logic factored by verb-object functions that interacted with a database system at the logical schema (e.g. relational) level. A typical order entry system built around a relational database management system (RDBMS) 20 years ago would have logic partitioned around verb-object functions associated with how users interacted with the system. In fact, the user interaction model via "screens" or "forms" became the primary factoring for logic – there would be a new-order screen, and update-customer screen. The system may have also supported batch updates of SKU's, inventory, etc. The application logic was tightly bound to the logical relational schema.

Much of the data-centric logic (e.g. validation logic) is embedded within the application logic. People typically wrote batch programs to interact directly with the logical schema to perform updates. Programming languages did not support representation of high-level abstractions directly – objects did not exist. These applications can be characterized as being closed systems whose logical data consistency was maintained by application logic implemented at the logical schema level. An order was an order because the new-order logic ensured that it was.

A key reason for custom data-centric logic by applications is the well-known ***application impedance mismatch problem***. The logical schema does not match the level of abstraction of the application. Applications address this problem by developing at the data abstraction (e.g. relational) and by writing custom mapping code to bridge the gap between the application and the data abstractions. This not only leads to duplication of effort but also reduces application development productivity. In the next sections we will show how the Entity Framework and the Language Integrated Query innovations in .NET languages help to minimize this impedance mismatch.

Several significant trends have shaped the way that modern data-based applications are factored and deployed today. Chief among these are object oriented factoring, service level application composition, and higher level data services. When we think about the factoring, composition, and services from above, we can see that the conceptual entities are an important part of today's applications. It is also easy to see how these entities must be mapped to a variety of representations and bound to a variety of services. There is no one correct representation or service binding. XML, Relational and Object representations are all important but no single one will suffice.

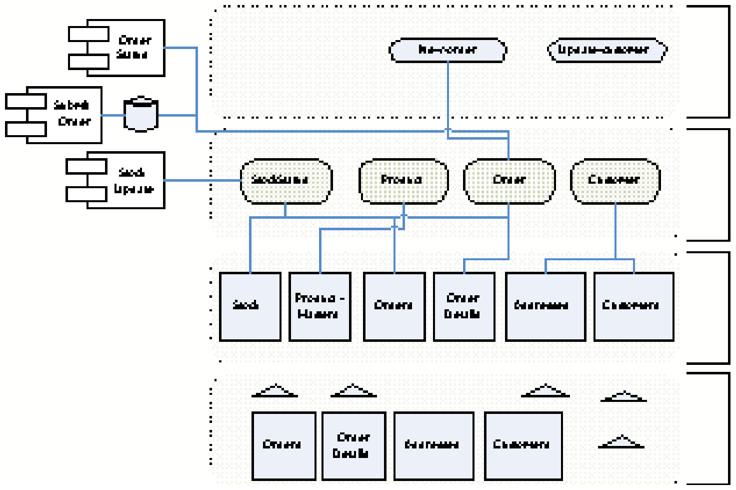


Figure 2 Order Entry System circa 2005

Consider a "StockNotifications" application which deals with concepts like Customer Order, Product, and Stock. How do we make them real and use our conceptual understanding of them throughout the system whether they are stored in a multi-dimensional database for analytics, in a durable queue between systems, in a mid-tier cache; a business object, etc.

Figure 2 captures the essence of this issue by focusing on several entities in our order entry system. Note that conceptual level entities have become real. Also note that the conceptual entities are communicating with and mapping to various logical schema formats, e.g. relational for the persistent storage, messages for the durable message queue on the Submit Order service, and perhaps XML for the Stock Update and Order Status web services.

3.2 SQL Server Evolution

The data services provided by a "data platform" 20 years ago were minimal and focused around the logical schema in an RDBMS. These services included query & update, atomic transactions, and bulk operations such as backup and load/extract.

SQL Server itself is evolving from a traditional RDBMS to a *complete data platform* that provides a number of high value data services over entities realized at the conceptual schema level. While providing services such as reporting, analysis, and data integration in a single product and realizing synergy among them was a conscious business strategy, the means to achieve these services and the resultant ways of describing the entities they operate over happened more organically – many times in response to problems recognized in trying to provide higher level data services over the logical schema level. There are two good examples of the need for concrete entity representation for services now provided within SQL Server: *logical records* for merge replication, and the *semantic model* for report builder.

Early versions of merge **replication** in SQL Server provided for multi-master replication of individual rows. In this early mode, rows can be updated independently by multiple agents; changes can conflict; and various conflict resolution mechanisms are provided with the model. This row-centric service had a fundamental flaw – it did not capture the fact that there is an implicit consistency guarantee around entities as they flow between systems. To address this flaw, the replication service introduced "logical records" as a way to describe and define consistency boundaries across entities comprised of multiple related rows at the logical schema level. "Logical records" are defined in the part of the SQL catalog associated with merge replication. There is no proper design-time tool experience to define a "logical record" such as an Order that includes its Order Details – applications do it through a series of stored procedure invocations.

Report Builder (RB) is another example of SQL Server providing a data service at the conceptual entity level. Since it operates at the logical schema level though, writing reports requires knowing how to compose queries at the logical schema level – e.g. creating an order status report requires knowing how to write the join across the several tables that make up an order. End users and analysts, however, want to write reports directly over Customers, Orders, Sales, etc. Thus, the SQL Server team created a means to describe and map conceptual entities to the logical schema layer we call the Semantic Model Definition Language (SMDL).

These are just two of a number of mapping services provided within SQL Server – the Unified Dimensional Model (UDM) provides a multi-dimensional view abstraction over several logical data models. A Data Source View (DSV), on which the BI tools work, also provides conceptual view mapping technology.

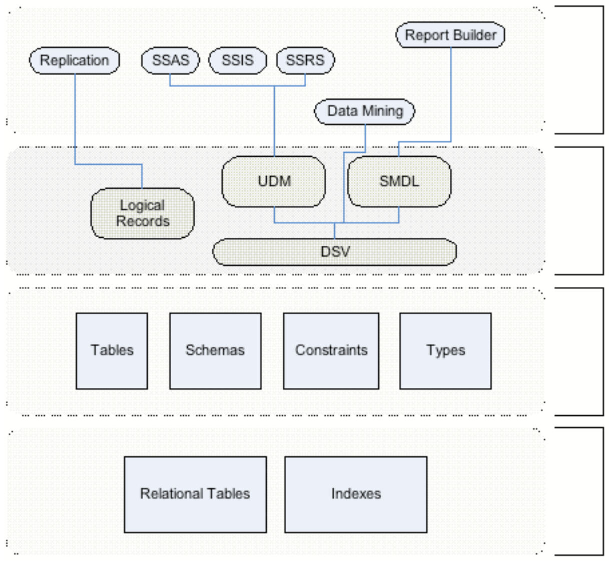


Figure 3: SQL Server 2005

A key observation is that several higher-level data services in the SQL Server product are increasingly delivering their services at the conceptual schema level. Currently, each of these services has a separate tool to describe conceptual entities and map them down to the underlying logical schema level. **Figure 3** illustrates the evolution of SQL Server into a data platform with many high value data services and multiple means to map conceptual entities to their underlying logical schemata.

4. ENTITY FRAMEWORK

This section describes the ADO.NET Entity Framework that makes the conceptual level real. We start with the rationale that led us to the development of an Entity Data Model (EDM) followed by an overview of the EDM. We present an architectural description of the entity framework implementing a runtime supporting the EDM, a query language, and mapping. We conclude the section with a description of the development process around the EDM.

4.1 Why a new model?

The Entity Data Model (EDM) is intended for developing rich data-centric applications. The obvious question that arises is: “why not use (or extend) one of these established data models? There are at least four other modern candidates for such a data model:

- The SQL data model (tables, columns, keys, referential integrity constraints...). SQL99 extends this core model to include object relational features (user defined-, structured-, and distinct-types, methods, typed tables, refs…).
- The CLR data model (classes, fields, methods, properties, value, and Ref types, collections…)
- The XSD model based on XML Infoset (Atomic-, list-, and union-types, primitive- and derived-types, token, ID, IDREF, ENTITY…)
- The UML data model (classes, objects, associations, generalizations, attributes, operations, aggregations…)

The overall reason is that we need something that maps cleanly to both the CLR and to relational databases like SQL Server, for programmability and persistence respectively. None of the other candidates has all the needed facilities for both. The CLR is an object-oriented, imperative-programming runtime, and has no native data model or notions of integrity constraints, relationships, or persistence. SQL99 lacks data modeling concepts like relationships, and does not have good programming language integration. The XSD specification does not support concepts like keys, relationships, and persistence. In addition, the full XSD specification is complex and has awkward mapping to both the runtime and to relational database models. The UML is too general: it requires application developers to add precise semantics, especially for persistence.

The EDM has been designed to map downward cleanly to both the CLR and to a relational database, and upward to a specialization of UML. Designers can work with concepts familiar from UML, which can be compiled in phases to XML, CLR programs, and SQL.

An important aspect of EDM is that it is value based like the relational model (and SQL) rather than object/reference based like C# (CLR). One or more object programming models can be easily supported on top of EDM. Similarly, the EDM can mapped to one or more relational DBMS implementations for persistence.

4.2 EDM Overview

The EDM extends the classic relational model with concepts from E-R modeling. The central concepts in the EDM are entities and relationships. *Entities* represent top-level objects with independent existence and identity, while *Relationships* are used to *relate* (or, describe relationships between) two or more entities.

4.2.1 Types

An ***EntityType*** describes the definition of an entity. An entity typically is a top-level object with independent existence. An entity has a *payload* - zero or more properties that describe the structure of the entity. Additionally, an entity type must define a *key* – a set of properties whose values uniquely identify the entity instance within its container. EntityTypes may derive from (or subtype) other entity types. EDM supports a single inheritance model.

The properties of an entity may be simple or complex types. A ***SimpleType*** (or a PrimitiveType) represents scalar (or atomic) types (e.g. integer, string), while a ***ComplexType*** can be used to represent structured properties (e.g. an Address). A ComplexType is composed of zero or more properties, which may themselves be scalar or complex type properties.

A ***Relationship*** type is a specialized entity type that describes relationships between two (or more) entity types. Initially, the EDM supports one kind of relationship, namely ***Association***, which models peer-to-peer entity relationships (e.g., Supplier-Part). Containment parent-child relationships (e.g. Order-Line) are modeled as associations with cascading actions. The key for a relationship type is usually, but not necessarily, the concatenated keys of the entity types participating in the relationship. Relationships – especially many-to-many relationships - may optionally contain properties of the relationship itself.

EDM ***Schemas*** provide a grouping mechanism for types – types must be defined in a schema.

In addition to the types above, the EDM supports *transient* types in the form of RowTypes and CollectionTypes. These occur mostly in the context of query operations (e.g., projections, joins). A *RowType* is an anonymous type that is structurally similar to a *ComplexType***.** A RowType's structure depends on the sequence of typed and named members that it is comprised of. A rowtype has no identity and cannot be inherited from. Instances of the same row type are equivalent if the corresponding members (in order) are respectively equivalent. Rows have no behavior beyond their structure. A *CollectionType* represents a homogenous collection of objects.

4.2.2 *Primitive Types*

The EDM is a data model, not a type system. The EDM defines shaping constructs (entity types etc.), but the actual types (and their semantics) are defined by the hosting environment. The EDM does define a set of abstract (or template) primitive types, and a set of associated facets, that enable the abstract primitive types to represent primitive types of the hosting environment (SqlServer databases, the CLR, etc.). These abstract types are proxies for the real primitive types defined by the host, and the semantics of operations over these types are entirely governed by the host.

4.2.3 *Instances*

Entity instances (or just entities) are logically contained within an ***EntitySet***. An EntitySet is a homogenous collection of entities (i.e.) all entities in an EntitySet must be of the same (or derived) EntityType. An entity instance must belong to exactly one entity set. In a similar fashion, relationship instances are logically contained within a ***RelationshipSet***. The definition of a RelationshipSet scopes the relationship, that is, it identifies the EntitySets that hold instances of the entity types that participate in the relationship. SimpleTypes and ComplexTypes can only be instantiated as properties of entity instances.

An ***EntityContainer*** is a logical grouping of EntitySets and RelationshipSets – akin to how a Schema is a grouping mechanism for EDM types.

4.2.4 *Examples*

```
<?xml version="1.0"?>
<Schema Namespace="CNorthwindSchema"
        xmlns="urn:schemas-microsoft-com:windows:storage">
<!–
Typical Entity definition, has identity and some members
-->
  <EntityType Name="Product" Key="ProductID">
    <Property Name="ProductID" Type="System.Int32" />
    <Property Name="ProductName" Type="System.String"
          Size="max" />
    ...
  </EntityType>

<!–
A derived product
-->
  <EntityType Name="DiscontinuedProduct" BaseType="Product">
    <Property Name="DiscReason" Type="System.String"
          Size="max" />
  </EntityType>

<!–
A complex type defines structure but no identity. It can be
used inline
in 0 or more Entity definitions
-->
   <ComplexType Name="CtAddress" >
      <Property Name="Address" Type="System.String"
          Size="max" />
      <Property Name="City" Type="System.String"
          Size="max" />
      <Property Name="PostalCode" Type="System.String"
          Size="max" />
      ...
   </ComplexType>
<!–
A Customer Entity
-->
   <EntityType Name="Customer" Key="CustomerID">
          <!– Address is a member which references a
          complextype -->
      <Property Name="Address" Type="CNorthwind.CtAddress" />
      <Property Name="CustomerID" Type="System.String"
           Size="max" />
   </EntityType>
<!–
An example of an association between Product [defined above]
and
OrderDetails [not shown for sake of brevity]
-->
  <Association Name="Order_DetailsProducts">
    <End Name="Product" Type="Product" Multiplicity="1" />
    <End Name="Order_Details" Type="OrderDetail"
          Multiplicity="*" />
  </Association>

</Schema>

<!–
The Entity Container defines the logical encapsulation of
EntitySets (sets of (possibly) polymorphic instances of a
type) and
AssociationSets (logical link tables for relating two or more
entity instances)
-->
  <EntityContainer Name="CNorthwind">
    <Using Namespace="CNorthwindSchema" />

    <EntitySet Name="Products" EntityType="Product" />
    <EntitySet Name="Customers" EntityType="Customer" />
    <EntitySet Name="Order_Details"
          EntityType="OrderDetail" />
    <EntitySet Name="Orders" EntityType="Order" />

    <AssociationSet Name="Order_DetailsProductsSet"
          Association="Order_DetailsProducts">
      <End Name="Product" EntitySet="Products" />
      <End Name="Order_Details" EntitySet="Order_Details"/>
    </AssociationSet>
  </EntityContainer>
```

4.3 Entity Framework Architecture

This section briefly describes the architecture of the Entity Framework being built as part of ADO.NET. The main functional components of the ADO.NET Entity Framework (see Error! Reference source not found.) are:

Data source-specific providers. The Entity Framework builds on the ADO.NET data provider model. There are specific providers for several relational, non-relational, and Web services sources.

EntityClient provider. The Entity Framework includes a new data provider, the EntityClient provider. This provider houses the services implementing the mapping transformation from conceptual to logical constructs. The EntityClient provider is a value-based, outside-the-store view runtime where data is accessed in terms of EDM entities and relationships and queried/updated using an entity-based SQL language (eSQL). The EntityClient provider includes the following services:

- **EDM/eSQL**. The EntityClient provider processes and exposes data in terms of the EDM values. Queries and updates are formulated using eSQL. They are processed through the query and update pipeline engines which incorporate mapping transformations and knowledge about the specific capabilities of the data sources.
- **Mapping**. View mapping, one of the key services of the EntityClient provider, is the subsystem that implements bidirectional (read and write) views that allow applications to manipulate data in terms of entities and relationships rather than rows and tables. The mapping from tables to entities is specified declaratively through a mapping definition language.
- **Store-specific bridge**. The bridge component is a service that supports the query execution capabilities of the query pipeline and coordinates the generation of queries using provider specific syntax.
- **Metadata services**. The metadata service supports all metadata discovery activities of the components running inside the EntityClient provider. All metadata associated with EDM concepts (entities, relationships, entitysets, relationshipsets), store concepts (tables, columns, constraints), and mapping concepts are exposed via metadata interfaces. The metadata services component also serves as a link between the domain modeling tools which support model-driven application design.
- **Transactions**. The EntityClient provider integrates with the transactional capabilities of the underlying stores.
- **API**. The API of the EntityClient provider follows the ADO.NET provider model based on Connection, Command, and DataReader objects. The EntityClient provider accepts commands in the form of eSQL text or canonical trees and produces DataReader objects as results.

Occasionally Connected Components. The Entity Framework enhances the well established disconnected programming model of the ADO.NET DataSet. In addition to enhancing the programming experiences around the typed and un-typed DataSets, the Entity Framework embraces the EDM to provide rich disconnected experiences around cached collections of entities and entitysets.

Embedded Database. The Entity Framework encompasses the capabilities of a low-memory footprint, embeddable database engine to enrich the services for applications that need rich middle-tier caching and disconnected programming experiences.

Design and Metadata Tools. The Entity Framework integrates with domain designers to enable model-driven application development. The tools include EDM, mapping, and query modelers.

Programming Layers. ADO.NET allows multiple programming layers to be plugged onto the value-based entity data services layer exposed by the EntityClient provider. The Object Services component is one such programming layer that surfaces CLR objects. There are multiple mechanisms by which a programming layer may interact with the entity framework. One of the important mechanisms is LINQ expression trees.

Services. Rich SQL data services such as reporting, replication, business analysis will be built on top of the Entity Framework.

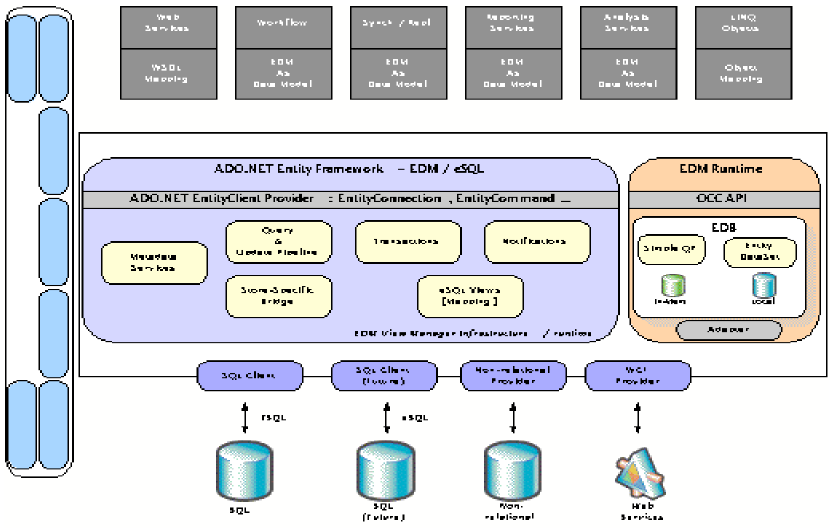


Figure 4 Entity Framework Architecture

4.4 Making the Conceptual Level real

This section outlines how one may define a conceptual model and work against it. We use a modified version of the Northwind database for familiarity.

4.4.1 Build the conceptual model

The first step is to define one's conceptual model. The EDM allows you to describe the model in terms of entities and relationships. The model may be defined explicitly by hand writing the XML serialized form of the model as shown above. Alternately, a graphical EDM designer tool may be used.

4.4.2 *Apply the mapping*

After we define the EDM conceptual model, we identify a target store, and then map the conceptual model to the target store's logical schema model. As with the conceptual EDM, one can hand write an explicit mapping or use a mapping tool. For example, the Northwind store may stripe data across multiple tables (the vertical partitioning strategy); however, applications would want to reason about the data as a single entity without the need for joins or knowledge of the relational model. The mapping layers isolate the application from knowledge of the store's schemas.

4.4.3 *Automatically Generated Classes*

Having the conceptual level is indeed sufficient for many applications as it provides a domain model that is live within the context of a comfortable pattern (ADO.NET commands, connections and data readers) and allows for late bound scenarios. Many applications, however, prefer an object programming layer (See **Figure 5**). This can be facilitated through code generation driven from the EDM description. For increased flexibility and data independence between the object and conceptual level, a mapping may be defined between classes and the conceptual model. The mapping between classes and the conceptual model is a straightforward member-wise mapping. This enables applications built against these classes to be reused against other versions of the conceptual model, provided a legal map can be defined.

4.4.4 *Using Objects*

One can interact with objects and perform regular Create, Read, Update and Delete (CRUD) operations on the objects. The example below demonstrates the use of Language Integrated Query (LINQ) to identify all orders that are newer than a given date

```
class DataAccess
{
 static void GetNewOrders(DateTime date) {
   using (NorthWindDB nw =
               new NorthWindDB ()) {
     var orders = from o in nw.Orders
               where o.OrderDate > date
               select new {o.orderID, o.OrderDate,
                         Total = o.OrderLines.Sum(
                  l => l.Quantity);

foreach (SalesOrder o in orders) {
       Console.WriteLine("{0:d}\t{1}\t{2}",
          o.OrderDate, o.OrderId, o.Total);
     }
   }
}
```

4.4.5 *Using Values*

There are many ISVs, framework and data services developers who just prefer to work against a .NET data provider; the EntityClient Provider is intended for such usage scenarios. The EntityClient Provider has a connection and a command and returns a DbDataReader when one invokes EntityCommand.ExecuteReader(). An example of a query using the EntityCommandCommand is as follows:

```
public void DoValueQueries(DateTime date)
{
 using (EntityConnection conn =
               new EntityConnection (connString))
 {
  conn.Open();
  EntityCommand command =
               conn.CreateCommand();
  command.CommandText =
               @"select value e from Employees as e
     where e.HireDate > @HireDate";
  command.Parameters.Add(
               new EntityParameter ("HireDate",
                         date));
  DbDataReader reader =
               command.ExecuteReader();
  while(reader.Read()) {
    //--- process record
  }
 }
}
```

5. SUMMARY AND CONCLUSION

Significant application and database technology trends require richer services at the conceptual rather than at the logical schema level. The Entity Framework provides a broad data platform with a rich and concrete conceptual schema to enable new applications and data services. The data platform includes the following components:

1. Entity Framework. A value-based runtime that implements an extended relational model - EDM - that embraces entities and relationships as first class concepts, a query language for the EDM, and a comprehensive mapping engine from the conceptual to the logical (relational) level.
2. Comprehensive programming model. We need programming model innovations that bridge the gap between different data representations (XML, relational, objects). In fact, by developing programming languages and APIs at the conceptual level, we will be able to liberate the programmer from the impedance mismatches that exist among different logical models. Programming language extensions such as Linq [5] provide richer, declarative programming models across different data representations.

3. Data services targeting the conceptual level. Examples include Synchronization/ Replication, Reporting, and Security.

4. Design-time tools. Data modeling tools today produce models that are largely abstract. They are used sometimes to produce a logical or physical design for a relational database implementation. We envision design-time tools that are used to: (a) build EDM models, (b) map EDM models to logical (relational) as well as other programming and presentation representations, and (c) semantics tools where you may introduce synonyms, aliases, translation and other semantic adornments for natural language and end user query.

6. ACKNOWLEDGMENTS

We would like to thank all members of the ADO.NET team for their contributions to building this system.

7. REFERENCES

[1] Chen, P. *The Entity-Relationship Model—toward a unified view of data*, ACM Transactions on Database Systems, Vol. 1, Issue 1, March 1976, pp. 9-36.

[2] Unified Modeling Language. http://www.uml.org/.

[3] Microsoft. The ADO.Net Entity Framework Overview. http://msdn.microsoft.com/data/default.aspx?pull=/library/en-us/dnvs05/html/ADONETEnFrmOvw.asp, June 2006.

[4] Blakeley, J.A., Campbell, D., Gray, J., Muralidhar, S., Nori, A.. Next-Generation Data Access: Making the Conceptual Level Real. http://msdn.microsoft.com/data/default.aspx?pull=/library/en-us/dnvs05/html/nxtgenda.asp, June 2006.

[5] Microsoft. The Linq Project. http://msdn.microsoft.com/data/ref/linq/default.aspx.

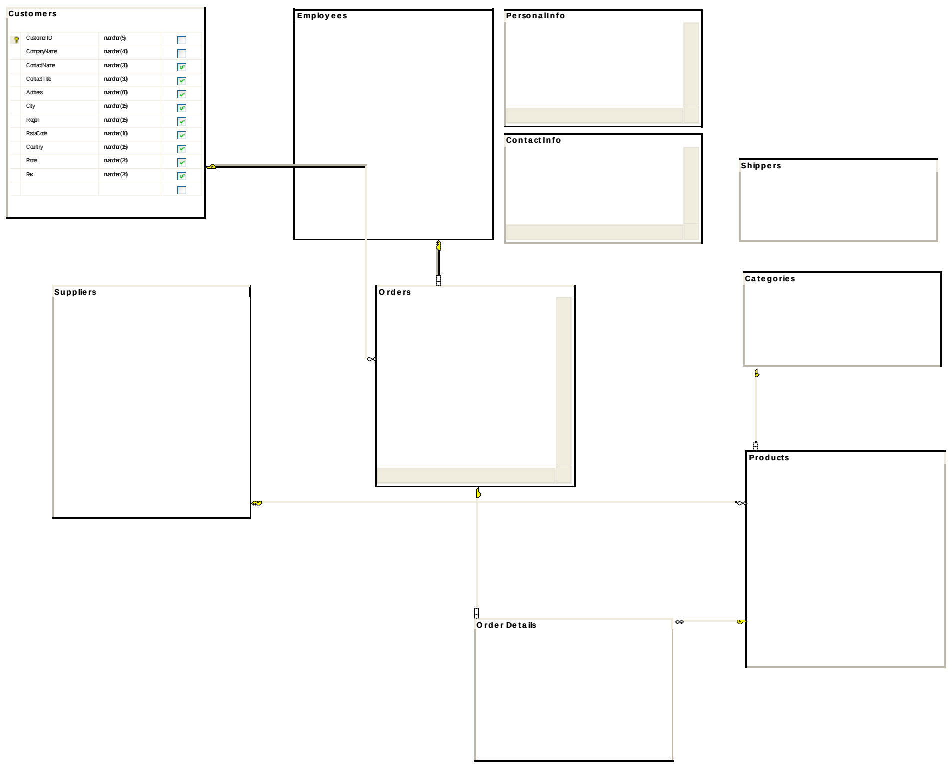


Figure 5: Entity Data Model for Northwind.

Data Management for a Smart Earth - The Swiss NCCR-MICS initiative -

Karl Aberer
School for Computer and Communication Science
EPF Lausanne
1015 Lausanne, Switzerland
+41 21 693 4679
karl.aberer@epfl.ch

Gustavo Alonso Donald Kossmann
Department of Computer Science
ETH Zurich
8092 Zurich, Switzerland
+41 44 632 (7306) / (2940)
{alonso, kossmann}@inf.ethz.ch

ABSTRACT

The Swiss National Competence Center for Research in mobile Information and Communication Systems (NCCR-MICS or MICS) is one of several research initiatives sponsored by the Swiss National Science Foundation to promote long term research projects in areas of vital strategic importance for the evolution of science in Switzerland, for the country's economy and for Swiss society. NCCR-MICS covers a wide spectrum of topics in the area of mobile information and communication systems ranging from information theory related to ad-hoc sensor networks to business models for pervasive computing, including network and routing issues, software and application development, and actual deployments of sensor networks (from architecture to geology). In this paper, we briefly present MICS as a whole and discuss in some detail two ambitious projects in the area of data management. The first project, XTream, addresses the whole life cycle of sensor based applications from the acquisition by sensors, aggregation and integration in gateways, storage in databases, generation of events that are relevant to users and applications, up to the subscription and consumption of events in a distributed architecture. The second project is Global Sensor Networks (GSN), which aims at enabling the rapid and efficient publication, sharing and interoperability of heterogeneous sensor data sources over large networks such as the Internet and P2P overlays.

1. INTRODUCTION

Wireless sensor networks are increasingly being used to monitor a wide range of natural phenomena and human activities. For instance, monitoring the watershed from glaciers to river mouths allows scientists to better understand the mechanisms governing the circulation of water and, thus, to improve prediction and management of this valuable resource. Similarly, wireless sensors and actuators in the walls of a building can be used to more efficiently control energy consumption while increasing the comfort of the users by creating individual microclimates. Wireless sensor networks are changing the way we use information technology: information becomes embedded into our physical environment by means of miniature devices and computers, providing dense sensing close to physical phenomena. This information is distributed, processed, stored, and fed into software applications that act on the information provided. The physical environment becomes thus intertwined with the Internet information space, evolving into what we call the *Smart Earth*.

Advances in technology are making it possible to acquire and store an ever increasing amount of data. Text and video are clear examples nowadays with individuals making vast amounts of information available and even producing it themselves now that the mechanisms exist to make it accessible to a broad audience. The key premise of our work is that sensor networks and sensor data will soon follow this trend. Hence, our goal is to provide dynamic mechanisms to cope with the resulting data deluge: rather than building large repositories with limited processing capacity, we focus on the publication, distribution, dissemination, and processing of sensor data.

This work is part of a large research initiative: the Swiss National Competence Center for Research in Information and Communication Systems (NCCR-MICS, *www.mics.ch*). The NCCR-MICS is tackling all technical aspects of sensor networks, from the study of fundamental principles (network structures, distributed algorithms, information and communication theory) to the development of platforms (wireless sensor technology, ad-hoc networks, in-network information processing, software verification), and their deployment in applications.

In this paper, we briefly present the NCCR-MICS, its structure and the broad research goals of its four clusters. Then, we concentrate on two concrete data management projects, *XTream* and *Global Sensor Networks* (GSN), and how they are addressing the data issues raised by the large scale deployment of sensor networks.

2. NCCR MICS

2.1 Structure and Organization

The NCCR MICS is a nation wide research center encompassing more than 40 faculty members across different Swiss universities and more than 90 Ph.D. students, mainly from Computer Science and Electrical Engineering but also with students in mechanical engineering, architecture, and business departments. MICS is currently on its second 4 year phase (from 2005 to 2009) after finishing a successful first phase from 2001 to 2005.

MICS is run by a Management Committee that acts as link between the project participants and the Swiss National Science Foundation. The management committee is assisted in this task by an external scientific board of 10 internationally renowned researchers that actively review MICS activities once a year. MICS is officially reviewed once a year by the Swiss National Science Foundation through a panel of 10 international experts. MICS also counts on an Advisory Board that consults on higher level strategic issues and is composed of five people with ample management experience in university, research and /or business administration. An open scientific conference is organized every 6 months at different locations in Switzerland where the work of all participants is presented in two or three intense days of keynotes, Ph.D. students' talks, posters, demonstration, panels, and strategic meetings[*]. These conferences coincide alternatively with the reviews of the scientific board and the Swiss National Science Foundation. There is also an annual summer school where external speakers are invited to give one week courses on a variety of topics related to the research areas covered in MICS.

MICS research is organized into 4 clusters:

- Theory of Self-organized, Distributed Communication and Information
- Mobile Communication and Processing Platforms
- Networked Software Systems
- In-network Information Management

Each cluster encompasses several research projects and one or more application projects. The application projects focus on real deployments of the technology developed within MICS.

2.2 MICS Cluster 1: Theory of self organized, distributed communication and information

Cluster 1 addresses the basic theoretical aspects of sensor and wireless networks in four areas: information theory, network theory, distributed signal processing and distributed algorithms. This cluster aims at establishing the basis for the work in all other clusters in that it develops the necessary fundamental understanding of the problems associated with sensor networks. The projects within this cluster explore a wide variety of issues from the trade-offs between data rates, reliability, bandwidth and energy consumption to mechanisms for signal reconstruction that can be used to accurately describe real phenomena using the data captured by a collection of sensors. This cluster also encompasses an application project in environmental monitoring, SensorScope, which aims at obtaining sensor measurements for supplying boundary conditions for complex physical models of environmental phenomena such as wind and water flow (*sensorscope.epfl.ch*).

2.3 MICS Cluster 2: Mobile Communication and Processing Platforms

Cluster 2 deals with technology that is required to cope with the challenges linked to the implementation and deployment of wireless ad-hoc networks: from basic routing to new technologies such as ultra-wide band communication. This cluster also explores two interesting applications. The first application involves using sensors to study the dynamics of rapid gravity-driven flows, such as avalanches and earth mass movements. The second application project builds distributed, self-organized, networked robotic olfactory systems for chemical plume mapping and odor source localization. Scientists involved in this cluster are also exploring how energy reduction can be achieved by using a more systematic cross-layer optimization, including additional upper and lower layers.

[*] Readers interested in attending or participating in these meetings are encouraged to contact any of the authors.

2.4 MICS Cluster 3: Networked Software Systems

Cluster 3 is devoted to the basic support necessary to develop applications that rely on sensor networks. The work in this cluster includes, among others, techniques to check the properties of software modules, using a combination of compile-time (off-line) and run-time (dynamic) analysis; the analysis of multi-threaded programs to detect errors or to issue warnings; exploring alternative architectures for sensor networks to facilitate deployment, monitoring and debugging; and proving communication protocols to be correct and secure. This cluster includes two application projects which deploy sensors in difficult-to reach regions: the PermaSense project (*cn.cs.unibas.ch/projects/permasense/*) focuses on the permafrost region in the Swiss alps (including placement of sensors on vertical mountain walls), the other project's focus is on water and humidity monitoring and management in an arid part of the Indian subcontinent.

2.5 MICS Cluster 4: In-Network Management Systems

Cluster 4 aims at supporting end-to-end data management for sensor and mobile networks covering all system layers and processing levels. The work in this cluster addresses the dire need for better tools (both actual as well as conceptual tools) to deal with the data generated by sensor networks. A common theme in this cluster is the use of a "declarative middleware" language and the interpretation of sensors as services. This cluster also takes a broader view on what a sensor is and generalizes sensors to any form of pull or push-enabled data source. The two projects that are described in the next two sections (XTream and GSN) are part of Cluster 4. Additional projects in this cluster address topics such as algorithms and mechanisms for the on-line detection of events in sensor networks (as opposed to just gathering data), protocols for the efficient dissemination of information across sensor networks, efficient mechanisms for concurrent programming in embedded systems, and studies on the commercial aspects of sensor networks. This cluster also has its own application project led by a group of architects that are trying to exploit sensor networks in buildings as a way to minimize construction and renovation costs as well as optimizing energy consumption in buildings.

3. XTREAM

3.1 Overall Goals

The XTream project is a collaborative effort within the Department of Computer Science of ETH Zurich. Its main goal is to look at the complete acquisition, distribution, delivery, and processing chain of data from sensor networks, understanding sensors in the widest possible sense. XTream targets all levels of the system from the software support at the sensor platforms, the middleware at the gateway of a sensor network, event generation and subscription, data stream processing, and declarative generation of data processing in distributed applications that must process diverse and heterogeneous data streams.

In its first phase, the XTream project tackled the limitations of existing programming platforms and languages to adequate them to the needs of sensor data processing. Again, these limitations appear at all levels and a significant effort has been made to define a coherent architecture with the proper abstractions and support at each level. In what follows, we present the results obtained so far.

3.2 SwissQM

One of the biggest limitations of existing sensor networks today is the lack of an adequate software platform to program the sensors themselves. What is currently available is either too low level (a stripped down operating system) and, thus, difficult to program, or too high level (query based systems) and, thus, not flexible enough. We ran into this problem at the very beginning when we started experimenting with simple sensor networks. The learning curve was very steap to program sensors, the resulting software was difficult to maintain, and high level systems simply did not support any of the things we wanted to do.

A first key contribution of XTream was SwissQM [4]: a flexible and extensible virtual machine that runs at the sensor nodes. SwissQM uses a bytecode language that is similar to Java bytecode with a few additions in order to be able to perform in-network data aggregation. SwissQM has been optimized for program size in order to minimize the overhead of distributing programs. It can concurrently run several programs, perform powerful data aggregation operations on-the-fly, and can be easily extended with user-defined functions. With SwissQM, it is trivial to, e.g., push down to the sensors data cleaning functionality like noise filters or window operators to minimize traffic and optimize the life time of the network. Being a programmable virtual machine, rather than, e.g., a SQL query engine, SwissQM is Turing complete and can be easily plugged into sophisticated software stacks that offer different interfaces to the outside world (including but not limited to SQL).

SwissQM is a key element in the XTream architecture because it abstracts the sensor hardware behind a simple bytecode language. Now we are in a position to develop front ends that provide either bridges to high level programming languages such as Java, XQuery and SQL or links to optimizers residing at higher layers in the architecture.

3.3 Gateway

A central component in the XTream architecture is a gateway server. The gateway server is more powerful than sensor nodes; it typically has high (wired) communication to the Internet and less limitations in terms of energy consumptions, main memory, and processing. The gateway collects data from the sensors and it compensates for functionality that cannot be implemented efficiently by the sensors directly. Furthermore, the gateway compiles SwissQM bytecode based on high-level declarative specifications, thereby carrying out optimizations and all planing that would be impossible to carry out by the individual sensors.

The gateway of XTream is the main processing engine of the sensor network. As with SwissQM, developing the gateway involved developing first adequate software support. The gateway is being implemented in Java on top of Concierge (*www.flowsgi.inf.ethz.ch/concierge.html*, now available as open source), an implementation of the OSGI specification tailored and optimized for small devices. Concierge allows the addition and removal of software modules at run time, a property that we will use to provide run-time extensibility. Through the gateway, we are building an SQL module, an XQuery module, and a web service module. What makes the gateway interesting is the fact that it can take the requests arriving through the different modules and apply multi-query optimization to the whole set. Through the gateway, we have demonstrated the ability to run more than a hundred concurrent user queries on a single sensor network by applying traditional optimization techniques [3].

3.4 SLETs and channels

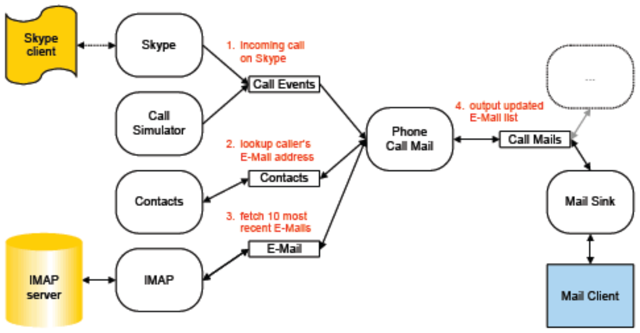


Figure 1. A first implemented XTream prototype combining streams from Skype and from an IMAP e-mail server

The ultimate goal of XTream is to create a development platform that will allow non specialists to create applications based on data streams. The concept is similar to mashups and built on similar primitives such as Web services. The architecture of XTream is divided into two components: Slets and channels (Figure 1). Slets are intended as the processing components that receive data, potentially from many channels, and produce data, potentially for many channels. Slets can be written in any programming language (e.g., Java); a declarative way to specify Slets is described in the next subsection. Channels are physical entities that abstract from the specifics of the underlying network, transport protocols, and data sources, offering a typed, XML based, query driven interface to mitigate the (potential) impedance mismatch between the programming language at the Slets and the data representation of the streams. They will also hide aspects such as access to remote data, refresh rates, access to multiple sources, etc.

Together, Slets and channels are used in order to compose complex applications from data streams and databases: Slets are the active components that encapsulate functionality and channels implement the dataflow between these active components. As a first approximation, the composition takes place in a workflow manner with a simple, yet powerful GUI that allows designers to plug channels into Slets and vice versa. These workflows are declarative and can be optimized in several different ways by the underlying runtime platform. An example of such an optimization is pushing down of operations from Slets into the gateway and possibly the sensor network; thanks to the design of SwissQM and the gateway such optimizations can indeed be implemented.

3.5 Programming Model

As mentioned in the previous subsection, XTream adopts a two-step programming model: Orchestration is carried out at the Slet/channel level and basic functionality is specified inside Slets, the gateway, and in the sensors themselves. For orchestration, any programming language can be used in order to program, say, Slets and any kind of data (e.g., comma separated values or RSS) can be shipped through channels. For best performance, however, XTream proposes extensions of XQuery in order to program Slets, the gateway, and even specify the operations that need to be carried out at the sensor level. XQuery is used for several reasons: First, XQuery makes it possible to process most data formats including, of course, comma-separated values and RSS. Second, the XQuery data model is based on sequences of items which is a perfect match for data streams. Third, XQuery is declarative, thereby enabling optimization and development tools (e.g., graphical editors and debuggers).

Unfortunately, the current XQuery standard is not powerful enough for XTream. Therefore, we are currently working on the following extensions, mostly in collaboration with Oracle and other industrial partners:

- Windows: This extension makes it possible to express tumbling windows, sliding windows, and landmark windows in XQuery in the same spirit as CQL (and other dialects) do for SQL. Window queries are particularly important at the gateway in order to carry out, e.g., data cleaning.
- Event generation: This extension makes it possible to compare two states of a database and notify users of critical state transitions (e.g., temperature dropped by more than ten degrees in ten minutes).
- Scripting: This extension adds error handling, variable assignment, and external function calls (e.g. Web service calls) to XQuery in order to allow more general programming [2].

3.6 Digital home scenario

As a proof of concept, and in collaboration with Siemens AG, we have developed a prototype that uses XTream to disseminate, process, control and display on a variety of hand-held devices information about a digital home (Figure 2).

The digital home scenario involves several house-related data streams (e.g., ringing of the door bell, storm warnings, status of home appliances) and personal data streams (e.g., E-Mails, information on phone calls, SMS, calendar events). All these data streams need to be processed in quasi-real-time, but with different requirements and priorities.

The prototype illustrates the goals and architecture of XTream. The different data sources are wrapped as slets that output data to channels. The channels are declaratively specified and the system forwards the data from slets to the relevant channels. Additional slets were developed in order to implement functionality such as merging streams, filtering out important events, or raising alarms when certain conditions occur. These slets forward their data to other channels that can be used again by new slets. User interfaces are hidden behind slets that read from channels. Certain slets control appliances; e.g., turn off the lights if all inhabitants have left the house. In this way, heterogeneity of both sources and sinks is hidden behind a uniform slet interface. The channels take care of the transport and distribution, as well as of storage of the data in transit. The Xtream software operates in a lab (a prototype house with real appliances). For testing purposes, Siemens provided a simulator, a configurable virtual digital home, as shown in Figure 2.

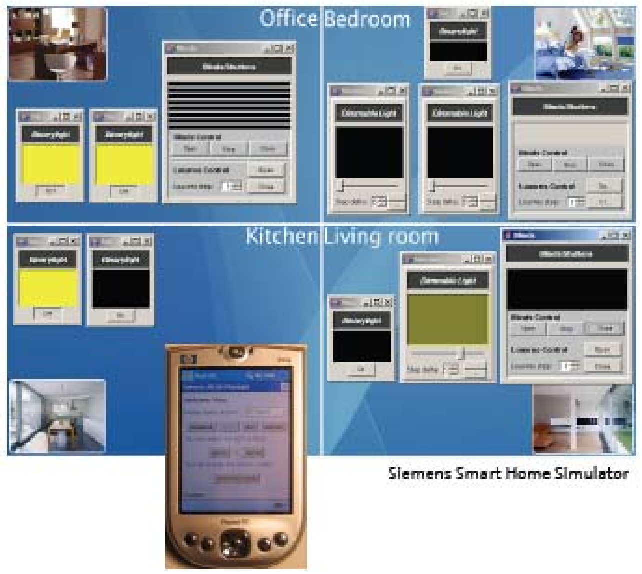


Figure 2. Simulator of a Digital Home controlled by the XTream prototype (in collaboration with Siemens)

4. GLOBAL SENSOR NETWORKS (GSN)

4.1 Overall Goals

As sensor network technology advances and the price of sensor networks rapidly diminishes, we can expect large numbers of sensor networks being deployed. This implies interesting opportunities and challenges for managing and sharing data produced by sensor networks at a global scale. Today we lack tools that would allow for rapid and efficient deployment of diverse sensor networks and for reuse and sharing of data generated by sensor networks at a global scale, despite of the similarity of the main tasks of processing, storing, querying and publishing data produced by a sensor network. The goal of Global Sensor Networks (GSN) is to provide a middleware platform that facilitates these tasks [1]. As a result, we expect to support developers of sensor networks in the rapid development of their applications and the simple publication of the data generated, and we expect to provide users an environment in which they can explore the sensor data space and potential applications in a way similar to the use of the current Internet.

In the following we provide an overview of the design considerations and features of a first system that has been developed recently and is being made available to the community as an open source release over Sourceforge (http://globalsn.sourceforge.net/)

The Global Sensor Networks (GSN) middleware provides a uniform platform for fast and flexible integration and deployment of heterogeneous sensor networks. The design of GSN follows four main design goals: Simplicity by using a minimal set of powerful abstractions which can be easily configured and adopted, adaptivity by enabling runtime reconfiguration when adding new types of sensor networks and data processing tasks, scalability and autonomy by using a peer-to-peer architecture, and light-weight implementation by ensuing a small memory foot-print, low hardware and bandwidth requirements, and web-based management tools.

4.2 Virtual sensors as Key Abstraction

A small set of powerful, easily combinable abstractions are key to successful middleware design. The key abstraction in GSN is the virtual sensor. Virtual sensors abstract from implementation details of access to sensor data and they are the services provided and managed by GSN. A virtual sensor corresponds either to a data stream received directly from sensors or to a data stream derived from other virtual sensors. A virtual sensor can have any number of input streams and produces one output stream. The specification of a virtual sensor provides all necessary information required for deploying and using it, including metadata used for identification and discovery, the structure of the data streams which the virtual sensor consumes and produces, a declarative SQL-based specification of the data stream processing performed in a virtual sensor, and functional properties related to persistency, error handling, life-cycle management, and physical deployment. To support rapid deployment, these properties of virtual sensors are provided in a declarative deployment descriptor specified in XML.

4.3 Data Stream Processing

In GSN a data stream is a sequence of timestamped tuples. The order of the data stream is derived from the ordering of the timestamps and the GSN container provides basic support to manage and manipulate the timestamps. These services essentially consist of the following components:

- a local clock at each GSN container
- implicit management of a timestamp attribute
- implicit timestamping of tuples upon arrival at the GSN container at reception time
- a windowing mechanism which allows the user to define count- or time-based windows on data streams.

In this way it is always possible to trace the temporal history of data stream elements throughout the processing history. Multiple time attributes can be associated with data streams and can be manipulated through SQL queries. In this way inherent properties of the observation process, such as network and processing delays, are made visible to applications for building their specific temporal abstractions on top of the available temporal data.

The production of a new output stream element of a virtual sensor is always triggered by the arrival of a data stream element from one of its input streams. Informally, the processing steps then are as follows:

- By default the new data stream element is timestamped using the local clock of the virtual sensor provided that the stream element had no timestamp.
- Based on the timestamps for each input stream the stream elements are selected according to the definition of the time window and the resulting sets of relations are unnested into flat relations.
- The input stream queries are evaluated and stored into temporary relations.
- The output query for producing the output stream element is executed based on the temporary relations.
- The result is permanently stored if required and all consumers of the virtual sensor are notified of the new stream element.

Additionally, GSN provides a number of possibilities to control the temporal processing of data streams, for example, bounding the rate of a data stream in order to avoid overloads of the system which might cause undesirable delays, sampling of data streams in order to reduce the data rate, and bounding the lifetime of a data stream in order to reserve resources only when they are needed.

GSN's query processing approach is related to TelegraphCQ (*telegraph.cs.berkeley.edu/telegraphcq/*) as it separates the time-related constructs from the actual query. Temporal specifications, e.g., the window size, are provided in XML in the virtual sensor specification, while data processing is specified in SQL. At the moment GSN supports SQL queries with the full range of operations allowed by the standard syntax, i.e., joins, subqueries, ordering, grouping, unions, intersections, etc. The advantage of using SQL is that it is well-known and SQL query optimization and planning techniques can be directly applied.

4.4 GSN architecture

GSN follows a container-based architecture and each container can host and manage one or more virtual sensors concurrently. The container manages every aspect of the virtual sensors at runtime including remote access, interaction with the sensor network, security, persistence, data filtering, concurrency, and access to and pooling of resources. This paradigm enables on-demand use and combination of sensor networks. Virtual sensor descriptions are identified by user-definable key-value pairs which are published in a peer-to-peer directory so that virtual sensors can be discovered and accessed based on any combination of their properties, for example, geographical location and sensor type. GSN nodes communicate among each other in a peer-to-peer fashion. Figure 3 depicts the internal architecture of a GSN node.

The virtual sensor manager (VSM) is responsible for providing access to the virtual sensors, managing the delivery of sensor data, and providing the necessary administrative infrastructure. Its life-cycle manager (LCM) subcomponent provides and manages the resources provided to a virtual sensor and manages the interactions with a virtual sensor (sensor readings, etc.) while the input stream manager (ISM) manages the input streams and ensures stream quality (disconnections, unexpected delays, missing values, etc.). The data from/to the VSM passes through the storage layer which is in charge of providing and managing persistent storage for data streams. Query processing is controlled by the query manager (QM) which includes the query processor being in charge of SQL parsing, query planning, and execution of queries (using an adaptive query execution plan). The notification manager deals with the delivery of events and query results to the registered clients. The top three layers deal with access to the GSN container.

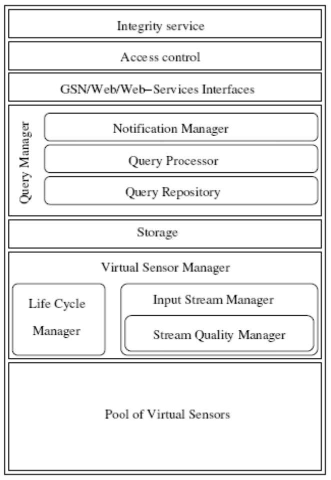


Figure 3. GSN architecture

4.5 Implementation

The GSN implementation consists of the GSN-CORE, implemented in Java, and the platform-specific GSN-WRAPPERS, implemented in Java, C, and C++, depending on the available toolkits for accessing sensors. The implementation currently has approximately 20,000 lines of code and is available from SourceForge. GSN is implemented to be highly modular in order to be deployable on various hardware platforms from workstations to small programmable PDAs, i.e., depending on the specific platforms only a subset of modules may be used. GSN also includes visualization systems for plotting data and visualizing the network structure.

For deploying a virtual sensor the user only has to specify an XML deployment descriptor as briefly outlined in Section 4.2 if GSN already includes software support for the concerned hardware and software. Adding a new type of sensor or sensor network can be done by supplying a Java wrapper conforming to the GSN API and interfacing the system to be included.

The effort to implement wrappers is quite low, i.e., typically around 100-200 lines of Java code. For example, the TinyOS wrapper required 150 lines of code. Our experience shows that new wrappers can be included usually in less than 1 day. Currently GSN includes already wrappers for the TinyOS family of motes (Mica, Mica2, Mica2Dot, TinyNodes, etc.), USB and wireless (HTTP-based) cameras (e.g., AXIS 206W camera), and several RFID readers (e.g., Texas Instruments).

The GSN implementation is highly performant. As an indication, the processing time for one virtual sensor deployed on a GSN node is approximately 0.1ms on a standard workstation. Thus, in performance evaluations we would typically host hundreds of virtual sensors on the same GSN node.

5. ACKNOWLEDGMENTS

NCCR MICS is a large collaboration effort in which the ideas of a large number of people have converged and been merged into a coherent whole. This paper reflects many of these shared ideas and, thus, we would like to thank all past and present participants of MICS for the collective and individual input. Regarding XTream, we would like to thank Irina Carbus, Michael Dueller, Tim Kraska, Rene Mueller, Christian Reichel, Jan Rellermeyer, Rokas Tamosevicius. Regarding GSN, we would like to thank Ali Salehi and Manfred Hauswirth. For information on the MICS projects, publications, pointers to all participants and an extensive list of publications and activities, consult the project web page: *www.mics.org*

The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems NCCR-MICS, a center supported by the Swiss National Science Foundation under grant number 5005-67322.

6. REFERENCES

[1] Karl Aberer, Manfred Hauswirth, Ali Salehi. **A middleware for fast and flexible sensor network deployment.** *32nd International Conference on Very Large Data Bases (VLDB2006), Seoul, Korea, 12-15 Sep 06*

[2] Don Chamberlin, Michael Carey, Daniela Florescu, Donald Kossmann, Jonathan Robie. **XQueryP: Programming with XQuery.** *3rd International Workshop on XQuery Implementation, Experience, and Perspectives (XIME-P2006). Chicago, Illinois, USA. 30 Jun, 2006.*

[3] Rene Mueller, Gustavo Alonso. **Efficient Sharing of Sensor Networks.** *IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS2007), Vancouver, Canada, 9-12 Oct 2006.*

[4] Rene Mueller, Gustavo Alonso, Donald Kossmann. **SwissQM: Next generation Data Processing in Sensor Networks.** *3rd Biennial Conference on Innovative Database Research (CIDR2007). Asilomar, California, USA. 7-10 Jan 07.*

Report on the Second International Workshop on Data Management on Modern Hardware (DaMoN'06)

Anastassia Ailamaki
Carnegie Mellon University
natassa@cmu.edu

Peter Boncz
CWI
boncz@cwi.nl

Stefan Manegold
CWI
manegold@cwi.nl

Abstract: This report summarizes the presentations and discussions that occurred during the Second International Workshop on Data Management on Modern Hardware (DaMoN). DaMoN was held in Chicago on June 25th, 2006, and was collocated with ACM SIGMOD 2006. The aim of this one-day workshop is to bring together researchers interested in optimizing database performance on modern computing infrastructure by designing new data management techniques and tools.

1 Motivation and Goal

The continued evolution of computing hardware and infrastructure imposes new challenges and bottlenecks to program performance. As a result, traditional database architectures that focus solely on I/O optimizations increasingly fail to utilize hardware resources efficiently. CPUs with superscalar out-of-order execution, simultaneous multi-threading, multi-level memory hierarchies, and future storage hardware (such as e.g. MEMS) impose a great challenge to optimizing database performance. Consequently, exploiting the characteristics of modern hardware has become an important topic of database systems research.

Architecture-conscious database research aims to make database systems adapt automatically to the sophisticated hardware characteristics, thus maximizing performance transparently to applications. To achieve this goal, the data management community needs interdisciplinary collaboration with computer architecture researchers. It also involves rethinking traditional data structures, query processing algorithms, and database software architectures to adapt to the advances in the underlying hardware infrastructure.

The International Workshop on Data Management on Modern Hardware (DaMoN) aims to achieve this goal by attracting people from computer architecture towards the principle data management venue ACM SIGMOD/PODS, both by providing a publication platform as well as through direct invitation, for the keynote talk and panel discussions.

2 Logistics and Statistics

The workshop was held on Sunday June 25th, the day before SIGMOD/PODS started. We had a program committee of 8 expert members from both areas of computer architecture and databases. There were eleven papers submitted to the workshop, which were professionally reviewed by the program committee (at least 3 reviews per submission). This number of submissions only just coincided with our minimum requirements, and may have been caused by overlap with other workshops (in particular, ExpDB) or by the unavoidable proximity of the submission deadline to that of VLDB. However, the submission quality was high, as measured by the absolute scores by the referees. With DaMoN in its second year, we achieved a 50% acceptance ratio.

Despite two round-of-sixteen world cup soccer matches, around 35 people attended DaMoN, making it one of the two best attended SIGMOD workshops (only WebDB had more attendees). This may also have to do with our keynote speaker (Berni Schiefer, IBM fellow) and a panel that included Mark Hill (professor of computer architecture from University of Wisconsin), Goetz Graefe (Microsoft CLR) and Todd Walter (CTO of NCR/Teradata), as well as Anastassia Ailamaki (database faculty at CMU). A striking characteristic considering attendance is that a considerable number of visitors came expressly to Chicago for DaMoN – which we interpret as testament to the relevance of the workshop for a community. Off the program committee Shimin Chen, Goetz Graefe and Bradley Kuszmaul attended the workshop. Logistically, everything went well in Chicago, including a nice lunch for a reasonable price. We specifically thank Kevin Chang, Joanne Martiori, Goce Trajcevski, and Lisa Singh for their help with the organization.

3 Technical Presentations

The six paper presentations were distributed into two sessions. In the first session, a research team by IBM described a solution for data mining on multiple information sources pertaining to different organizations, while respecting data privacy and without disclosing information to each organization. This was achieved by encrypted data transport and running the data mining algorithms inside a sealed tamper-protected hardware subsystem, that runs on an embedded processor. Scaling down data mining algorithms to fit the reduced resources available in such an environment was the main topic of this work. The second presentation stayed on the data mining topic, investigating the use of Processing-In-

Memory chips for various link-discovery algorithms, with varying degrees of success.

Last in this session was an architecture-conscious study of a large transaction benchmark set-up. This cooperation between Intel and Microsoft industry labs provides in-depth experimental data of running SQLserver on a parallel Itanium2 machine. This paper provides the best reference material of various hardware event traces (cache misses, branch misprediction, TLB handling) published so far on a realistic high-performance transaction processing scenario. Due to the expensive hardware setup (32 CPUs, 256 GB RAM, 1260 disks), such experiments are generally out of reach for academic researchers, but insights from these measurements can help set their agendas. As organizers, we highly value papers like these.

In the second session, a combination of researchers from Sandia Labs and Columbia University experimented with the massive multi-processors reality that awaits the database community within a decade. Rather than relying on simulation, they used an existing Cray MTA-2 super computer to test a variety of processing strategies for join and selection. This well-written and informative paper was awarded the Best Paper Award, collected by John Cieslewicz in behalf of fellow authors Ken Ross, Jonathan Berry and Bruce Hendrickson.

The remaining two presentations focused on architecture-conscious index structures. In the first, Goetz Graefe addressed the problem that in B-tree search with wide nodes, binary search inside each node will cause significant cache misses. Interpolation search can reduce their number, but is non-trivial to make robust against different data distributions. His paper presents a comprehensive overview of techniques and strategies for robust use of interpolation search in B-trees. The final presentation described the use of the new Cuckoo Hash technique in databases. Like a perfect hash function, Cuckoo obviates the need for a collision list, and thus reduces loop dependencies during hash-lookup, without requiring in-advance knowledge of the data distribution. This last paper also describes an adaptive buffering strategy in partitioned hashing, that reduces the RAM requirements of operators like hash aggregation and join, but conserves their CPU cache efficiency.

4 Panel Discussion

The concluding panel "ManyCore-DB: will you still need me, will you still feed me, when I'm 64?" discussed the influences of the computer architecture trend towards massively (e.g. 64-way) multi-core CPUs on data management.. The panel, moderated like last year by Peter Boncz, once again turned out to be very lively.

The first speaker was Mark Hill, who provided an excellent overview from the computer architect's view of the near-future parallel reality, underlining the renewed relevance that the topic of parallelism should receive in the next years in our research community. Next up was Goetz Graefe, who is currently working on support for parallelism in the Common Language Runtime (CLR) at Microsoft. CLR has been in the database picture for the recent developments around LINQ, that bring certain database processing models right into programming language environments. Goetz was, however, tight-lipped regarding any as-yet-unannounced Microsoft product features. As the sole representative of the database academia, Natassa Ailamaki discussed the challenges of massive CPU parallelism and made a case for staged database architecture to address these. Last up was Todd Walter, the CTO of NCR/Teradata. He provided many insights, predicting among others the demise of multiple-socket architectures (SMP) once massively multi-core CPUs become the norm. But, he pointed out that in computer architecture, the main bottleneck is I/O and the continued exponential growth in computational power provided by parallelism will not be matched by similar advances in magnetic storage. The panelists, however, discarded any known storage mechanism (MEMS, Flash) as a potential successor to magnetic disks.

The overall conclusion was that massive multi-core parallellism will create massive under-utilization of processing power very soon. This outcome we view as highly supportive for the future relevance of architecture-conscious data management research.

5 Where do we go from here?

This second edition of DaMoN has shown that the workshop can consistently draw 30-40 attendees, with a considerable number of submissions and visitors coming from outside the database field. The event also draws significant attention from industry both in submissions and attendees, and this year was generously sponsored by Intel Research. The quality of the papers presented has been good, exemplified by a previous DaMoN paper in extended form winning the Best Paper at VLDB 2005. As a result, the organizers plan to continue DaMoN also at SIGMOD 2007 in Beijing.

A Report on the Eighth ACM International Workshop on Data Warehousing and OLAP (DOLAP'05)

Juan Trujillo
Dept. of Language and Information Systems
University of Alicante, Spain
Apto. Correos 99. E-03080
{jnmazon,jtrujillo}@dlsi.ua.es

Il-Yeol Song
Drexel University
Colleageue of Information Science & Technology
Philadelphia, PA 19104
songiy@drexel.edu

1. Introduction

Research in data warehousing and OLAP has produced important technologies for the design, management and use of information systems for decision support. Much of the interest and success in this area can be attributed to the need for software and tools to improve data management and analysis given the large amounts of information that are being accumulated in corporate as well as scientific databases. However, in spite of the maturity of these technologies, new data needs or applications currently run at companies not only demand more capacity, but also new methods, models, techniques or architectures to satisfy these new needs.

This report focuses on the *Eighth ACM International Workshop on Warehousing and OLAP (DOLAP'05)* held in conjunction with the 14th International Conference on Information and Knowledge Management (CIKM'05) in Bremen, Germany, on November, 1st-5th, 2005. A summary of the accepted papers, the invited keynote speaker and the Panel discussions held at the end of the workshop is given.

In the call for papers, papers focused on new research directions and emerging application domains in the areas of data warehousing and OLAP were especially encouraged. In response to the call for papers, the workshop received 31 submissions from 18 different countries and only 12 papers were selected by the Program Committee, making an acceptance rate of 38.7%.

The accepted papers were organized in five different sessions: (i) querying OLAP databases, (ii) data warehouse models, (iii, iv) data warehouse design, and (v) query processing and view maintenance.

2. Querying OLAP Databases

Pu [1] focuses on the problem of representing OLAP databases and their query language. To this aim, the author first defines a framework based on functional symbols annotated by typing information. Then, once the basic multidimensional database has been defined, query constructs are specified as higher-order polymorphic functions, and queries are expressed as complex functional expressions. The author argues that its query language is flexible enough to represent useful OLAP queries as well as new user defined functions can also be easily expressed. Moreover, the presented equational specifications can help designers to automatically reason about problems such as summarizability of OLAP views. Finally, Pu argues that static type analysis of OLAP queries has not been dealt with existing formalisms.

Ladjel *et al.* [2] argue that in most cases, end users cannot properly query the DW mainly because (i) sometimes the most interesting information is not correctly shown, and (ii) the OLAP answers provided by the corresponding OLAP tools cannot be properly visualized. For this reason, the authors present a framework in which end users (i) specify their information preferences by ordering the different parts of an OLAP query (e.g. dimensions, classification hierarchy levels, and so on), and (ii) define their visualization constraints (mainly imposed by the limitations of the device used for the OLAP queries). The authors argue that their framework takes into account both personalization and visualization at the same time.

3. Data Warehouse Models

Perez *et al.* [3] start by arguing that many data contexts of current corporate data warehouses can be found in both external and internal documents. Thus, the authors propose a Relevance-Extended Multidimensional Model to combine structure of DWs and documents in what they call Data Warehouse Contextualized with Documents. To this aim, authors propose the *R-cube*, a type of OLAP cube based on (i) specifying the relevance of each fact in a query, and (ii) defining the related documents that provide information on the selected facts. In this way, users can query a traditional DW (with the corresponding MD terms) and obtain further information stored in related documents.

Jones and Song [4] claim that designing dimensional models can be complex, costly and time consuming. For this reason, they apply well-known software pattern techniques in the developing dimensional models. Authors identify and classify main dimensional patterns that normally occur in specifying dimensions. Then, they provide a metamodel to help designers in specifying Dimensional Design Patterns (DDP) and apply it to the design of dimensional models. The initial results of the experiment reveal a significant increase in the efficiency of their students in designing dimensional models.

Bimonte *et al.* [5] present a multidimensional model in order to correctly represent spatial data for data warehouses. Spatial data need complex structures that cannot be correctly managed by traditional data warehouses. Thus, the authors claim that supporting and exploiting the particular nature of spatial data into the MD analysis implies the re-thinking of basic OLAP concepts, such as spatial dimensions or facts, or the logical and physical representation of these spatial dimensions and facts. Therefore, the authors propose a Spatial Multidimensional Model that allows us to represent complex (spatial) objects following the MD paradigm in order to handle geographical data. They show that spatial data warehouses can be modeled based on the traditional conception of the MD paradigm.

4. Data Warehouse Design

Giorgini *et al.* [6] claim that a significant percentage of data warehouses fails to meet business objectives. The authors argue that requirement analysis is typically overlooked in real world DW projects. In this paper, they propose a goal-oriented approach to requirement analysis for DWs, based on the Tropos methodology. The authors integrate in the same analysis requirement approach both (i) organizational modeling, centered on stakeholders, and (ii) decisional modeling, focused on decision makers. The authors argue that their approach can be used in both a demand-driven and a mixed supply/demand-driven design framework. Finally, the authors provide some snapshots of the prototype that implement their methodological approach and apply it to a real case study.

Mazon *et al.* [7] argue that most existing modeling approaches do not provide designers with an integrated and standard method for designing the whole DW (ETL processes, data sources, DW repository and so on). In this work, the authors present a novel approach to align the whole DW development process to MDA (Model Driven Architecture). Then, they focus on the MD analysis and define the MD2A (Multidimensional Model Driven Architecture) as an approach for applying the MDA framework to MD modeling. They first describe how to build the different MDA artifacts (i.e. models) by using extensions of the Unified Modeling Language (UML). Then, transformations between models are clearly and formally established by using the Query/View/Transformation (QVT) approach. Finally, the authors provide an example on how to apply MDA and its transformations to the MD modeling.

Simitsis [8] describes the mapping of conceptual models into logical models for ETL processes. The author starts by identifying how a conceptual entity is mapped to a logical entity. Next, he determines the execution order in the logical workflow by using information adapted from the conceptual model. Finally, he provides a methodology for the transition from the conceptual to the logical model.

Nguyen *et al.* [9] start by arguing that traditional business intelligence (BI) architectures lack the support of real-time BI and closed-loop decision making. In this work, authors present a real-time BI architecture called SARESA. The main aim of the BI architecture is to provide continuous, real-time analytics in order to enable proactive responses to a business environment for effectively managing and controlling time-sensitive business processes. Then, the authors describe the *Sense & Respond* loops and a service-oriented architecture that is able to detect situations and exceptions, perform complex analytical tasks and reflect on the gap between current situations and desired management goals. Finally, they apply it to a mobile phone fraud detection case study.

5. Query Processing and View Maintenance

The work presented by Dehne *et al*. in [10] relies on supporting efficient indexing facilities for M cube queries. The authors argue that the complexity and difficulty of the indexing problem is exacerbated by the existence of attribute hierarchies that sub-divide attributes into aggregation layers of varying granularity. Thus, they present a hierarchy and caching framework that supports the efficient and transparent manipulation of attribute hierarchies within a parallel ROLAP environment. They also provide experimental results that verify that very little overhead is required to handle streams of arbitrary hierarchical queries.

Cuzocrea's work [11] focuses on the efficient execution of approximate answers for OLAP applications. The author claims that the scalability of the query techniques and the accuracy of the answers are recognized as important limitations of state-of-the-art approximate query answering proposals in OLAP. In this paper, Alfredo presents a statistical framework that covers the limitations related to the accuracy of queries. This is done by ensuring the probabilistic bounds on the retrieved answers, and tailored for the specific OLAP context. Within this framework, the author defines the KSyn synopsis data structure, which efficiently supports approximate query answering in OLAP. Finally, Alfredo presents and discusses encouraging preliminary experimental results stating the goodness of his proposal.

Lee and Kim [12] start by defining the multiple view maintenance problem and arguing that materialized views are still commonly used in data warehouse environments. The authors claim that although there has been much work on efficient maintenance of a single view, maintenance of multiple views has not been sufficiently investigated. Then, in this paper they propose an efficient incremental maintenance of multiple join views. Basically, they propose the delta propagation strategy that computes the change of multiple join views in a recursive manner. Then, the authors provide a heuristic algorithm that finds a global maintenance plan for the given views. Finally, they present an experimental result that shows the efficiency of the proposed method.

6. Keynote address

In DOLAP'05, we had a highly interesting keynote address entitled *"My Favorite Issues in Data Warehouse Modeling"* given by Jens Lechtenbörger [13]. He started by sketching major achievements and trends in conceptual DW modelling and pinpointed open problems along the way. Then, he took a closer look at the overall design process focusing on the transformation of conceptual data warehouse schemata into logical ones, and arguing that there is still a semantic gap between advanced conceptual data models and relational or MD implementations, which needs to be bridged. Finally, he turned to one specific aspect of the DW lifecycle, namely schema changes, and highlighted challenges in DW schema versioning.

7. Panel Discussions

At the end of DOLAP'05 we organized a Panel discussion among all participants and identified new research directions of data warehouses and OLAP technologies. Some ideas came out for possible future collaboration between different research groups. Some of the hot topics identified were DW security, data quality, visualization, ETL processes, distributed DWs, advanced OLAP for business intelligence, web warehouses, DWs for new applications such as XML documents, stream data, spatial or GIS data or biomedical data.

8. Conclusions / Summary

DOLAP'05 continued with the successful series of DOLAP workshops, being an international forum where both researchers and practitioners can share their findings in theoretical foundations, current methodologies, and practical experiences. As seen throughout this report, this year, DOLAP'05 was specially focused on new research directions and emerging application domains in the areas of data warehousing and OLAP. Thanks to the high quality of the presented papers, we were able to host a special issue by extending the best papers in the *Decision Support Systems* journal [15].

9. Acknowledgments

We would like to express our gratitude to the Program Committee members and the additional external referees for their hard work in reviewing papers. We also thank all the authors for submitting their papers and the CIKM2005 organizing committee for all their support. Another gratitude is for our technician Miguel A. Varo, who developed the web site and the review system (http://gplsi.dlsi.ua.es/congresos/dolap05/). This workshop was organized with the support from the following projects: METASIGN (TIN2004-00779), DADASMECA (GV05/ 220) and DADS (PBC-05-012-2).

10. References

[1] K. Q. Pu "Modeling, Querying and Reasoning about OLAP Databases: a Functional Approach", in [14].

[2] L. Bellatrach *et al.* "A Personalization Framework for OLAP Queries", in [14].

[3] J. M. Pérez *et al.* "A Relevance-Extended Multi-dimensional Model for a Data Warehouse Contextualized with Documents", in [14].

[4] M.E. Jones and I-Y. Song. "Dimensional Modeling: Identifying, Classifying, and Applying Patterns ", in [14].

[5] S. Bimonte *et al.* "Towards a Spatial Multidimensional Model", in [14].

[6] P. Giorgini *et al.* "Goal-Oriented Requirement Analysis for Data Warehouse Design", in [14].

[7] J-N. Mazón *et al.* "Applying MDA to the Development of Data Warehouses", in [14].

[8] A. Simitsis. "Mapping Conceptual to Logical Models for ETL Processes ", in [14].

[9] T. M. Nguyen *et al.* "Sense & Response Service Architecture (SARESA): An Approach Towards a Real-time Business Intelligence Solution and Its Use for a Fraud Detection Application", in [14].

[10] F. Dehne *et al.* "Parallel Querying of ROLAP Cubes in the Presence of Hierarchies", in [14].

[11] A. Cuzzocrea. "Providing Probabilistically-Bounded Approximate Answers to Non-Holistic Aggregate Range Queries in OLAP", in [14].

[12] K.Y. Lee and M.H. Kim. "Optimizing the Incremental Maintenance of Multiple Join Views", in [14].

[13] Jens Lechtenborger. "Issues in Data Warehouse Modeling", in [14]

[14] J. Trujillo, I-Y. Song (eds.). "Eighth ACM International Workshop on Data Warehousing and OLAP", ISBN: 1-59593-162-7, ACM Press, NY, 2005.

[15] *Decision Support Systems* journal. Ed: Elsevier. ISSN: 0167-9236. In press.

ICDE 2006 Ph.D. Workshop Report

Wai Gen Yee
Dept. of Computer Science
Illinois Institute of Technology
10 W. 31st Street
Chicago, IL 60616
yee@iit.edu

Shamkant B. Navathe
College of Computing
Georgia Institute of Technology
801 Atlantic Drive
Atlanta, GA 30332
sham@cc.gatech.edu

ABSTRACT

This report summarizes the Ph.D. Workshop held in conjunction with the 2006 IEEE International Conference on Data Engineering. This report includes a summary of the technical presentations as well as the panelist discussion.

1. INTRODUCTION

The goal of the 2006 IEEE International Conference on Data Engineering's Ph.D. Workshop is to increase the exposure of Ph.D. students to the conference and academic environments in a supportive and constructive way. This year's Workshop consisted of two parts: technical presentations of attendees' work as well as a panelist discussion. More details on the workshop, including papers and slides can be found on the Web at [1].

This structure of the report corresponds to the structure of the Workshop. Technical content and then panelist discussion are summarized in sequence.

2. SUMMARY OF TECHNICAL PROGRAM

The topics selected for the Workshop's technical program roughly mirrors the current research general research directions in data management. Contributors were also from several parts of the world, reflecting, perhaps, regional area focuses.

We divided the paper presentation into four sessions: Data Stream Management Systems, Query Processing, Data Mining and Semantics, and Distributed Systems. Because of the nature of the Workshop, some work may be ongoing, and we will describe outstanding issues where appropriate.

The first talk of the Data Stream Management System session was entitled *Control-Based Load Shedding in Data Stream Management Systems* by Yi-Chen Tu and Sunil Prabakar. This work shows how query accuracy can be maintained while reducing data loss by applying control theory and related feedback mechanisms.

The second talk of the Data Stream Management System session was entitled *Scalable and Adaptable Distributed Stream Processing* by Yonghuan Zhou. This work describes how a hierarchical architecture of streamed data workload coordinators that distribute load among stream data clients can be scalable, simple to implement, and cost-effective.

The third talk of the Data Stream Management System session was entitled *Processing High-Volume Stream Queries on a Supercomputer* by Eric Zeitler and Tore Risch. This talk described an experience of implementing a stream query processor on a supercomputer for the LOFAR system (a software radio telescope) and went into detail about how design decisions impacted cost and performance.

The first talk of the Query Processing session was entitled *Supporting Predicate-Window Queries in a Data Stream Management System* by Thanaa M. Ghanem. This work claims that the use of a "predicate window" over streaming data, which considers a streamed tuple being semantically equivalent to a traditional update of to a materialized view, is a more powerful generalization of existing "sliding window" operations on stream data. This semantic equivalence allows queries over the predicate window to be expressed using minimum extensions to SQL.

The second talk of the Query Processing session was entitled *Optimization of Complex Queries in Relational Databases* by Bin Cao. This talk described how SQL queries containing nested subqueries that are either aggregate or non-aggregate can be optimized by transformations that limit redundant computations.

The third talk of the Query Processing session was entitled *Twig Query Processing under Concurrent Updates* by Christian Mathis and Theo Harder. This work describes a technique for allowing high concurrency in an environment where XML data are being updated, subject to transactional properties, by using fine-grain locks.

The first talk of the Data Mining and Semantics session was entitled *A Generalized Framework for Mining Spatial and Spatio-temporal Patterns in Scientific Data* by Hui Yang and Srinivasan Parthasarathy. This talk described the application of data mining techniques to extract information scalably from geometric objects, possibly capturing temporal information, for scientific computing applications.

The second talk of the Data Mining and Semantics session was entitled *Searching and Ranking Documents Based on Semantic Relationships* by Boanerges Aleman-Meza. This talk described the use of named entities on documents and their semantic relationships to improve query result ranking, much in the same way link analysis is a structural means of improving result ranking.

The third talk of the Data Mining and Semantics session was entitled *Using Data Extraction Ontology to Foster Automating Semantic Annotation* by Yihong Ding and David W. Embley. This talk described an automated Web page annotator that works by using ontologies as an input to the data extraction process to help guide the annotation process and to make system adaptive to a large number of domains. Further, object extraction is integrated into this process to improve performance.

The fourth talk of the Data Mining and Semantics session was entitled *Model Video Semantics with Constraints Considering Temporal Structure and Typed Events* by Yu Wang, Lizhu Zhou, and Jianyong Wang. This talk described a way of modeling and constraining entities and events temporally in a way that is suitable for video annotation and querying.

The first talk of the Distributed Systems session was entitled *Location-based Spatial Queries with Data Sharing in Mobile Environments* by Wei-Shinn Ku and Roger Zimmermann. This talk describes a system for finding nearest neighbor queries in a mobile peer-to-peer environment. Each query uses partial results from other peers as well as the triangle inequality to estimate nearest neighbor results.

The second talk of the Distributed Systems session was entitled *MoSCoE: A Framework for Modeling Web Service Composition and Execution* by Jyotishman Pathak, Samik Basu, Robyn Lutz, and Vasant Honavar. This talk described a system for incremental composition of Web services that allows for a high degree of flexibility in case of partial failure of the composition.

The third talk of the Distributed Systems session was entitled *Ant Algorithms for Search in Unstructured Peer-to-Peer Networks* by Elke Michlmayr. This talk described a way of routing peer-to-peer queries in a random network by incrementally learning best paths by using a technique similar to pheromone trail blazing done by ants.

3. SUMMARY OF PANELIST DISCUSSION

The goal of the panelist session was two-fold: ask for technical advice on their work or ask about perspectives on career choices. As many of the students attendees were about to graduate, they focused almost exclusively on the latter goal. Furthermore, as most of the student attendees were interested in academic jobs, most questions focused on attaining and succeeding in an academic career.

The panelists were selected in an attempt to mix up seniority levels. They were:

- Laura Bright (post-doctoral researcher, Portland State University)
- Ramez Elmasri (professor, University of Texas, Arlington)
- Christopher Jermaine (assistant professor, University of Florida)
- Vijay Kumar (professor, University of Missouri, Kansas City)
- Wai Gen Yee (assistant professor, Illinois Institute of Technology)
- Sham Navathe (professor, moderator, Georgia Institute of Technology)

Students were first asked about how they would be judged when looking for an academic position. The criteria include (in no particular order) pedigree, publication record, letters of recommendation, demographic profile. Pedigree (i.e., having a degree from a top-ranked institution) indicates that a student has trained with the leaders in an area. Hiring someone with a good pedigree increases the chance for collaborations and associations between the respective institutions. On a more subtle note, the embarrassment on the part of the hiring institution of hiring someone with a good pedigree who ultimately fails is less than if that someone came from a less prestigious institution.

There is no doubt that a graduating student must publish. The question is whether it is the quality or the quantity of publications that matters. The consensus is that quality of publication is more important than quantity. Hiring committees want to know what the peak intellectual output of a person is, and assume that, given enough resources (e.g., equipment, students), publication quantity will happen.

Letter of recommendations are important to help hiring committees understand how much a particular student contributed to his/her stated research. For example, was the student the initiator of his/her research ideas? Was he the workhorse? Does s/he work well in teams? Does he understand the big picture? Hiring committees would likely avoid candidates who are not going to be able to produce results independently but will also avoid those who isolate themselves.

The demographic profile of a candidate is important as well. Although an institution will first search for scholarship and departmental need, all else being equal, an American university is likely to give preference to Americans and to underrepresented ethnic and gender groups. The justification for this is that it may introduce more perspectives into the environment and improve the educational potential of the department. Note that foreign applications are welcome at most institutions; however, many smaller institutions do not have the recruiting budget to invite such candidates.

A few of the students asked about the possibility of working in industry for a few years before returning to academia. Most of the panelists scoffed at this idea: it is simply too difficult, in most settings, to maintain a publication stream while working a full-time industry job.

4. CONCLUSIONS AND POSSIBLE DIRECTIONS OF FUTURE WORK

From our feedback from students, the ICDE06 Ph.D. Workshop was a success, having achieved its stated goals. Students particularly appreciated the ability to ask questions and get advice from Ph.D.s of various ranks from various backgrounds.

Much of the success was due to the contributions of the authors and how they were able to interact despite their disparate areas of specialization. We were also fortunate to have contributions from more senior members of the research community join various sessions and act as panelists. Their input made a significant impact.

Future Ph.D. Workshops may be improved by shortening each author's presentation time and having a poster session. This would allow more specialized student-to-student interaction as well as greater student participation, as we learned in organizing the Midwest Database Research Colloquium [2].

Furthermore, we also noticed that many of the questions posed during the panel session were the same questions that have been echoed in our personal advisement of students as well as when we were students. We believe that such questions are general to all Ph.D. students and believe that a repository of such questions and answers should be compiled as a student reference. Such a repository could, say, store the collected wisdom from other recent attempts at group advisement (e.g., [4]).

In effect, we could form a persistent community advisement system. Technically, a Web blog may be the appropriate means of implementing such a system (perhaps [3]). The success of such a system would depend on its broad support from the entire community. We would be happy to hear any feedback or pledges of support for such an idea.

5. ACKNOWLEDGMENTS

We would like to thank the paper contributors, the attendees, the program committee and the ICDE organizers for their support of the Workshop. We would also like to acknowledge Brian Cooper for his idea on community advisement of database students.

REFERENCES

[1] ICDE 2006 Ph.D. Workshop Web Site. http://ir.iit.edu/~waigen/icde06phd/.

[2] Midwest Database Research Symposium Web Site. http://dais.cs.uiuc.edu/mwdbrs/.

[3] DB Advisor Blog. http://dbadvisor.blogspot.com/.

[4] SIGMOD 2006 Life After Graduation Symposium Web Site. http://tangra.si.umich.edu/clair/sigmod-pods06/graduation.htm.

Report on the Models of Trust for the Web Workshop (MTW'06)

Tim Finin
University of Maryland, Baltimore County
Maryland MD 21250 USA
finin@umbc.edu

Lalana Kagal
Massachusetts Institute of Technology
Cambridge MA 02139 USA
lkagal@csail.mit.edu

Daniel Olmedilla
L3S Research Center and Hannover University
30539 Hannover Germany
olmedilla@l3s.de

1 Introduction

We live in a time when millions of people are adding information to the Web through a growing collection of tools and platforms. Ordinary citizens publish all kinds of content on Web pages, blogs, wikis, podcasts, vlogs, message boards, shared spreadsheets, and new publishing forums that seem to appear almost monthly. As it becomes easier for people to add information to the Web, it is more difficult, and also more important, to distinguish reliable information and sources from those that are not.

Search engines excel at finding results that are relevant to a user's query, but many are outdated, biased, inaccurate, and/or from unreliable sources. Popularity based metrics such as Google's PageRank help, but users are still forced to filter the results to select the most reliable information based on their particular trust requirements. With the introduction of web services, the problem is further exacerbated as users have to come up with a new set of requirements for trusting web services and web services themselves require a more automated way of trusting each other. Apart from inaccurate or outdated information, we also need to anticipate *Semantic Web spam* (SWAM) – where malicious sources publish false facts and scams to deliberately mislead software agents and programs.

The *Models of Trust for the Web* workshop was held in conjunction with the 15th International World Wide Web Conference (WWW2006) on 22nd May, 2006 in Edinburgh, Scotland. The goal of the workshop was to bring together researchers and experts from different communities (e.g., Information Systems, Database, Semantic Web, Web Services) who have been working on topics like trust, provenance, privacy, security, reputation, and spam, in order to understand the challenges associated with facilitating trust on the Web, to deliver a state-of-the-art overview in the area, and to identify guidelines for future research. The workshop built on several related workshops, including the *Workshop on Policy Management for the Web* [1] held at the 2005 World Wide Web conference and *Semantic Web and Policy Workshop* [2] held in conjunction with 2005 International Semantic Web Conference.

The MTW'06 workshop was attended by over thirty researchers for a full day of presentations, panels and spirited discussions. The eleven papers that were presented covered a wide spectrum of topics from inferring trust, to using trust to prevent spam, and the role of social networks in calculating trust [3].

2 Presented Papers

The keynote speaker, Ricardo Baeza-Yates (Yahoo! Research), discussed how social networks can be exploited to provide social and economic deterrents for spamming. There are several kinds of spam that need to be monitored: scraper scam that copies good data from other sites and adds monetization, synthetic text that provides boilerplate text built around key phrases, query-targeted spam in which each page targets a single tail query, DNS spam where many domains use the same servers, and blog spam. Using Flickr as an example, Ricardo showed how the "wisdom of crowds can be used to search" as Flickr users collaboratively search and tag each other's photos and the anchor text is collective knowledge used to create a search. At Yahoo!, spam is detected and characterized using a combination of algorithmic and editorial techniques in order to prevent it from distorting the ranking of web pages.

2.1 Session I : Trust Networks

David Brondsema and Andrew Schamp described their work in using social trust networks to filter spam in "Konfidi: Trust Networks Using PGP and RDF". They proposed that spam can be filtered by reasoning over trust relationships in RDF. These relationships include who (both identity and public key) is trusted, value of trust, and with respect to what topic.

In "Using Trust and Provenance for Content Filtering on the Semantic Web", Jennifer Golbeck and Aaron Mannes showed that annotating relationships with binary trust values in web-based social networks allowed trust values to be inferred between unrelated entities. They discussed the FilmTrust project, which is a movie recommendation system developed using their approach.

Patricia Victor et al. defined their billatice trust model that takes trust, distrust, lack of data, and contradictory data into consideration while calculating trust in "Towards a Provenance-Preserving Trust Model in Agent Networks".

2.2 Session II : Inferring Trust

The paper "Propagating Trust and Distrust to Demote Web Spam" by Baoning Wu, Vinay Goel, and Brian Davison addressed the problem of web spam also known as search engine spam in which a target page gets undeserved ranking. They described different methods that a parent page can use to divide its trust or distrust among its child pages. They also defined mechanisms for calculating trust (using outgoing links) and distrust (using incoming links) - accumulation, maximum share, and maximum parent.

L. Jean Camp, Cathleen McGrath, and Alla Genkina approached human trust behavior from a social science perspective. They described their results in "Security and Morality: A Tale of User Deceit" in which they present how users "consider failures in benevolence more serious than failures in competence".

Deborah McGuinness et al. reported in "Investigations into Trust for Collaborative Information Repositories: A Wikipedia Case Study" that both provenance of information and revision details are required to improve the trustworthiness of collaborative information systems such as Wikipedia. They discussed citation-based trust, which is derived from citation relationships among articles, and revision-based trust, which is derived from the original article, revision operators, and revision authors, as mechanisms for inferring trustworthiness of a Wikipedia article. They also presented a mockup of a Wikipedia version marked up with trust.

2.3 Session III : Trust Models

Santtu Toivonen, Gabriele Lenzini, and Ilkka Uusitalo explored the role of context in trust determination in their paper "Context-aware Trust Evaluation Functions for Dynamic Reconfigurable Systems". They distinguished between quality attributes, which are static attributes of the trustee and context attributes, which are optional attributes that can change dynamically such as location, and device type. They discussed how context-aware trust is calculated from quality attributes, reputation, and recommendations within a certain trust scope at a certain time.

In "How Certain is Recommended Trust-Information", Uwe Roth and Volker Fusenig suggested that trust information given by a recommender may not be reliable and could negatively affect the trust decision. They proposed a strategy for making trust decisions based on a converting a network of relations of direct and recommended trust information into a decision tree by choosing the path as well as whether to trust the recommended information along the path at random.

2.4 Session IV : Trust in Applications

The paper titled "Quality Labeling of Web Content: The Quatro approach" by Vangelis Karkaletsis et al. reports on a common machine-readable vocabulary for labelling web content that will be represented by user friendly icons for ease of understanding. The vocabulary includes several kinds of labels: page-specific such as whether the page uses clear language, whether it includes a privacy statement, content provider specific such as whether his credentials have been verified, business-specific such as whether it complies with rules and regulations of e-business, and label-specific such as when the label was issued, and when it was last viewed.

Ing-Xiang Chen and Cheng-Zen Yang examined biased search engine results and their deviations in "A Study of Web Search Engine Bias and its Assessment". An example of search engine bias is results in China for the keyword "Falun Gong". The authors proposed a two-dimensional scheme by adopting both indexical bias (differences in the sets of URLs retrieved) and content bias (deviations of content).

Alex Tsow suggested that users are prone to attacks from malicious software embedded on their routers in

2

"Phishing with Consumer Electronics - Malicious Home Routers" and hinted that trust is not a software only matter but there is an implicit trust in hardware vendors.

3 Future Directions / Open Research Issues

The *Models of Trust for the Web* workshop helped to understand current state-of-the-art research and to provide a discussion forum for researchers working on trust issues. It highlighted the importance of existing lines of research and brought up some salient emerging problems. Some open questions and problems include:

- *Trust modeling:* Is trust just boolean or does it have some certainty associated? Is it transitive? May we infer that the enemy of my enemy is my friend? Is it context or time dependent?
- *Trust awareness:* Users' trust on computers highly depends on previous experiences. They are initially too trustful and typically unaware of the risks associated with their computer usage. Improving user interfaces and increasing user awareness on privacy and security issues remains an open issue.
- *Trustworthy information:* Whether information retrieved from unknown sources is correct or not is a major question in current Web. Furthermore, SWAM makes this task even more difficult.
- *Database Security:* Trust in data and systems over time (compliance storage) is currently a major issue (specially after latest laws in which companies need to store data for longer time). The current winner of the best paper award at the Very Large Database Conference "Trustworthy Keyword Search for Regulatory-Compliant Record Retention" [4] demonstrates the importance of this line of research.
- *Access control & Trust Management:* Specification of conditions under which service access is granted or private information is released in an open distributed world has been and is still one of the most important lines of research. Work on policies is currently a hot topic in areas such as the Semantic Web [2, 5] for example.

4 Conclusion

The Web and its evolving infrastructure have made it easy to access virtually all of the world's knowledge and is the first source to which most of us turn when we need to know something. Search engines and other tools have focused on finding information *relevant* to users' queries with results ranked at best by their popularity. As it becomes easier to publish information on the Web, it is increasingly important to develop good frameworks for evaluating the trustworthiness of the information found. The papers from the WWW'06 *Models of Trust for the Web* workshop addressed these issues, identified important issues, and offered some partial solutions.

References

[1] Lalana Kagal, Tim Finin, and James Hendler, editors. *Proceedings of the WWW'05 Workshop on Policy Management for the Web*, Chiba JP, May 2005. http://ebiquity.umbc.edu/paper/html/id/221/.

[2] Lalana Kagal, Tim Finin, and James Hendler, editors. *Proceedings of the ISWC 2005 Semantic Web and Policy Workshop*, Galway IE, November 2005. http://ebiquity.umbc.edu/paper/html/id/268/.

[3] Tim Finin, Lalana Kagal, and Daniel Olmedilla, editors. *Proceedings of the WWW'06 Workshop on Models of Trust for the Web*, volume 190 of *CEUR Workshop Proceedings*, Edinburgh, Scotland, May 2006. CEUR-WS.org. http://ceur-ws.org/Vol-190.

[4] Soumyadeb Mitra, Windsor W. Hsu, and Marianne Winslett. Trustworthy keyword search for regulatory-compliant record retention. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 1001–1012, Seoul, Korea, Sep 2006.

[5] Piero A. Bonatti, Li Ding, Tim Finin, and Daniel Olmedilla, editors. *Proceedings of the ISWC'06 2nd International Semantic Web Policy Workshop (SWPW)*, Athens GA USA, November 2006.

Normalization Theory for XML*

Marcelo Arenas
Department of Computer Science
Pontificia Universidad Católica de Chile
marenas@ing.puc.cl

1 Introduction

Since the beginnings of the relational model, it was clear for the database community that the process of designing a database is a nontrivial and time-consuming task. Even for simple application domains, there are many possible ways of storing the data of interest.

During the 70s and 80s, a lot of effort was put into developing methodologies to aid in the process of deciding how to store data in a relational database. The most prominent approaches developed at that time – which today are an standard part of the relational technology– were the entity-relationship and the normalization approach. In the normalization approach, an already designed relational database is given as input, together with some semantic information provided by a user in the form of relationships between different parts of the database, called *data dependencies*. This semantic information is then used to check whether the design has some desirable properties, and if this is not the case, it is also used to convert the poor design into an equivalent well-designed database.

The normalization approach was proposed in the early 70s by Codd [9, 10]. In this approach, a *normal form*, defined as a syntactic condition on data dependencies, specifies a property that a well-designed database must satisfy. Normalization as a way of producing good relational database designs is a well-understood topic. In the 70s and 80s, normal forms such as 3NF [9], BCNF [10], 4NF [13], and PJ/NF [14] were introduced to deal with the design of relational databases having different types of data dependencies. These normal forms, together with normalization algorithms for converting a poorly designed database into a well-designed database, can be found today in every database textbook.

With the development of the Web, new data models have started to play a more prominent role. In particular, XML (eXtensible Markup Language) has emerged as the standard data model for storing and interchanging data on the Web. As more companies adopt XML as the primary data model for storing information, the problem of designing XML databases is becoming more relevant.

The concepts of database design and normal forms are central in relational database technology. In this paper, we show how these concepts can be extended to XML databases. The goal of this paper is to present principles for good XML data design. We believe this research is especially relevant nowadays, since a huge amount of data is being put on the Web. Once massive Web databases are created, it is very hard to change their organization; thus, there is a risk of having large amounts of widely accessible, but poorly organized legacy data.

Designing a relational database means choosing an appropriate relational schema for the data of interest. A relational schema consists of a set of relations, or tables, and a set of data dependencies over these relations. Designing an XML database is similar: An appropriate *XML schema* has to be chosen, which usually consists of a DTD (Document Type Definition) and a set of data dependencies. However, the structure of XML documents, which are trees as opposed to relations, and the rather expressive constraints imposed by DTDs make the design problem for XML databases quite challenging.

This paper is organized as follows. In Section 2, we show that XML documents may contain redundant information, which could be related to the hierarchical structure of these documents. In Section 3, we present the basic terminology used in this paper. In Section 4, we introduce a functional dependency language for XML, which is used in Section 5 to define XNF, a normal form for XML documents. In Section

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

6, we study the complexity of verifying whether an XML document is in XNF. In Section 7, we present an information-theoretic approach that can be used to justify normal forms and, in particular, XNF. We conclude the paper with some final remarks in Section 8.

2 Motivation: Redundant Information in XML

How does one identify bad designs? We have looked at a large number of DTDs and found two kinds of commonly present design problems. One of them is reminiscent of the canonical example of bad relational design caused by non-key functional dependencies, while the other one is more closely related to the hierarchical structure of XML documents, as we illustrate in the example below.

Example 2.1 Consider the following DTD that describes a part of a database for storing data about conferences.

```
<!ELEMENT db (conf*)>
<!ELEMENT conf (issue+)>
  <!ATTLIST conf
            title CDATA #REQUIRED>
<!ELEMENT issue (inproceedings+)>
<!ELEMENT inproceedings EMPTY>
  <!ATTLIST inproceedings
            title CDATA #REQUIRED
            pages CDATA #REQUIRED
            year CDATA #REQUIRED>
```

Each conference has a title, and one or more issues (which correspond to years when the conference was held). Papers are stored in `inproceedings` elements; the year of publication is one of its attributes.

Such a document satisfies the following constraint: any two `inproceedings` children of the same `issue` must have the same value of `year`. This too is similar to relational functional dependencies, but now we refer to the values (the `year` attribute) as well as the structure (children of the same `issue`). Moreover, we only talk about `inproceedings` nodes that are children of the same `issue` element. Thus, this functional dependency can be considered relative to each `issue`.

The functional dependency here leads to redundancy: year is stored multiple times for a conference. The natural solution to the problem in this case is not to create a new element for storing the year, but rather restructure the document and make `year` an attribute of `issue`. That is, we change attribute lists as:

```
<!ATTLIST issue
          year CDATA #REQUIRED>
<!ATTLIST inproceedings
          title CDATA #REQUIRED
          pages CDATA #REQUIRED>
```

□

Our goal is to show how to detect anomalies of those kinds.

3 Notation

We shall use a somewhat simplified model of XML trees in order to keep the notation simple. We assume a countably infinite set of labels L, a countably infinite set of attributes A (we shall use the notation $@a_1, @a_2$, etc for attributes to distinguish them from labels), and a countably infinite set V of values of attributes. Furthermore, we do not consider `PCDATA` elements in XML trees since they can always be represented by attributes.

A DTD (Document Type Definition) D is a 4-tuple (L_0, P, R, r) where L_0 is a finite subset of L, P is a set of rules $\ell \to P_\ell$ for each $\ell \in L_0$, where P_ℓ is a regular expression over $L_0 - \{r\}$, R assigns to each $\ell \in L_0$ a finite subset of A (possibly empty; $R(\ell)$ is the set of attributes of ℓ), and $r \in L_0$ (the root).

Example 3.1 The DTD shown in Example 2.1 is represented as (L_0, P, R, r), where $r = db$, $L_0 = \{db, conf, issue, inproceedings\}$, $P = \{db \to conf^*, conf \to issue^+, issue \to inproceedings^+, inproceedings \to \epsilon\}$, $R(conf) = \{@title\}$, $R(inproceedings) = \{@title, @pages, @year\}$ and $R(db) = R(issue) = \emptyset$. □

An XML tree is a finite rooted directed tree $T = (N, E)$ where N is the set of nodes and E is the set of edges, together with the labeling function $\lambda : N \to L$ and partial attribute value functions $\rho_{@a} : N \to V$ for each $@a \in A$. We furthermore assume that for every node x in N, its children $x_1, \ldots, x_n$ are ordered and $\rho_{@a}(x)$ is defined for a finite set of attributes $@a$. We say that T conforms to DTD $D = (L_0, P, R, r)$, written as $T \models D$, if the root of T is labeled r, for every $x \in N$ with $\lambda(x) = \ell$, the word $\lambda(x_1) \cdots \lambda(x_n)$ that consists of the labels of its children belongs to the language denoted by P_ℓ, and for every $x \in N$ with $\lambda(x) = \ell$, we have that $@a \in R(\ell)$ if and only if the function $\rho_{@a}$ is defined on x (and thus provides the value of attribute $@a$).

2

4 Functional Dependencies for XML

To present a functional dependency language for XML we need to introduce some terminology. An *element path* q is a word in L^*, and an *attribute path* is a word of the form $q.@a$, where $q \in L^*$ and $@a \in A$. An element path q is consistent with a DTD D if there is a tree $T \models D$ that contains a node reachable by q (in particular, all such paths must have r as the first letter); if in addition the nodes reachable by q have attribute $@a$, then the attribute path $q.@a$ is consistent with D. The set of all paths (element or attribute) consistent with D is denoted by $paths(D)$. This set is finite for a non-recursive D and infinite if D is recursive.

An *XML functional dependency (XFD)* over DTD D [4] is an expression of the form $\{q_1, \ldots, q_n\} \rightarrow q$, where $n \geq 1$ and $q, q_1, \ldots, q_n \in paths(D)$. To define the notion of satisfaction for XFDs, we use a relational representation of XML trees from [4]. Given $T = (N, E)$ that conforms to D, a *tree tuple* in T is a mapping $t : paths(D) \rightarrow N \cup V \cup \{\bot\}$ such that if q is an element path whose last letter is ℓ and $t(q) \neq \bot$, then

- $t(q) \in N$ and its label, $\lambda(t(q))$, is ℓ;
- if q' is a prefix of q, then $t(q') \neq \bot$ and the node $t(q')$ lies on the path from the root to $t(q)$ in T;
- if $@a$ is defined for $t(q)$ and its value is $v \in V$, then $t(q.@a) = v$.

Intuitively, a tree tuple assigns nodes or attribute values or nulls ($\bot$) to paths in a consistent manner. A tree tuple is maximal if it cannot be extended to another one by changing some nulls to values from $N \cup V$. The set of maximal tree tuples is denoted by $tuples_D(T)$. Now we say that XFD $\varphi = \{q_1, \ldots, q_n\} \rightarrow q$ is true in T, denoted by $T \models \varphi$, if for any $t_1, t_2 \in tuples_D(T)$, whenever $t_1(q_i) = t_2(q_i) \neq \bot$ for all $i \leq n$, then $t_1(q) = t_2(q)$ holds.

Example 4.1 Among the XFDs over the DTD from Example 2.1 one can find the following:

```
db.conf.@title → db.conf,
db.conf.issue →
        db.conf.issue.inproceedings.@year.
```

The first functional dependency specifies that two distinct conferences must have distinct titles. The second one specifies that any two *inproceedings* children of the same *issue* must have the same value of *@year*. □

Given a DTD D and a set $\Sigma \cup \{\varphi\}$ of XFDs over D, we say that (D, Σ) *implies* φ, written $(D, \Sigma) \vdash \varphi$, if for every tree T with $T \models D$ and $T \models \Sigma$, it is the case that $T \models \varphi$. The set of all XFDs implied by (D, Σ) is denoted by $(D, \Sigma)^+$. Furthermore, an XFD φ is *trivial* if $(D, \emptyset) \vdash \varphi$. In relational databases, the only trivial FDs are $X \rightarrow Y$, with $Y \subseteq X$. Here, DTD forces some more interesting trivial functional dependencies. For instance, for each element path p in D and p' prefix of p, $(D, \emptyset) \vdash p \rightarrow p'$, and $(D, \emptyset) \vdash p \rightarrow p.@a$. As a matter of fact, trivial functional dependencies in XML documents can be much more complicated than in the relational case, as we show in the following example.

Example 4.2 Assume that $r \rightarrow (a|b|c)$ is a rule in a DTD D, being r the type of the root. Then, for every path p in D, XFD $\{r.a, r.b\} \rightarrow p$ is trivial since for every XML tree T conforming to D and every tree tuple t in T, $t(r.a) = \bot$ or $t(r.b) = \bot$. □

We conclude this section by pointing out that other proposals for XFDs exist in the literature [16, 20, 28]. In particular, the language introduced in [28] is similar to the one presented in this paper.

5 XNF: An XML Normal Form

With the definitions of the previous section, we are ready to present a normal form for XML documents.

Definition 5.1 [4] *Given a DTD D and a set Σ of XFDs over D, (D, Σ) is in* XML normal form (XNF) *iff for every nontrivial XFD $X \rightarrow p.@a \in (D, \Sigma)^+$, it is the case that $X \rightarrow p$ is in $(D, \Sigma)^+$.*

The intuition is as follows. Suppose that $X \rightarrow p.@a$ is in $(D, \Sigma)^+$. If T is an XML tree conforming to D and satisfying Σ, then in T for every set of values of the elements in X, we can find only one value of $p.@a$. Thus, to avoid storing redundant information, for every set of values of X we should store the value of $p.@a$ only once; in other words, $X \rightarrow p$ must be implied by (D, Σ).

In this definition, we impose the condition that φ is a nontrivial XFD. Indeed, the trivial XFD $p.@a \rightarrow$

$p.@a$ is always in $(D, \Sigma)^+$, but often $p.@a \rightarrow p \notin (D, \Sigma)^+$, which does not necessarily represent a bad design.

To show how XNF distinguishes good XML design from bad design, we revisit our running example.

Example 5.2 The conference example 2.1 seen earlier may contain redundant information: year is stored multiple times for the same issue of a conference. This XML specification is *not* in XNF since

$$\texttt{db.conf.issue} \rightarrow \texttt{db.conf.issue.inproceedings.@year} \quad (1)$$

is a nontrivial XFD in the specification but

$$\texttt{db.conf.issue} \rightarrow \texttt{db.conf.issue.inproceedings}$$

is not in $(D, \Sigma)^+$, as several papers are usually published in a conference. The solution we proposed in the introduction was to make year an attribute of issue. XFD (1) is not valid in the revised specification, which can be easily verified to be in XNF. Note that we do not replace (1) by

$$\texttt{db.conf.issue} \rightarrow \texttt{db.conf.issue.@year},$$

since it is a trivial XFD and thus is implied by the new DTD alone. □

5.1 BCNF and XNF

In this section, we show that XNF generalizes BCNF. Recall that a relation specification (G, FD) is in BCNF, where relation G has attributes $A_1, \ldots, A_n$ and FD is a set of functional dependencies over G, if for every nontrivial FD $X \rightarrow Y$ implied by FD, we have that X is a superkey, that is, $X \rightarrow A_i$ is implied by Σ, for each $i \in [1, n]$.

Relational databases can be easily mapped into XML documents. Given a relation $G(A_1, \ldots, A_n)$ and a set of FDs Σ over G, we translate the schema (G, FD) into an XML representation, that is, a DTD D_G and a set of XFDs Σ_{FD}. The DTD $D_G = (L_0, P, R, db)$ is defined as follows: $L_0 = \{db, G\}$, $A = \{@A_1, \ldots, @A_n\}$, $P = \{db \rightarrow G^*,\ G \rightarrow \epsilon\}$, $R(db) = \emptyset$ and $R(G) = \{@A_1, \ldots, @A_n\}$. Without loss of generality, assume that all FDs are of the form $X \rightarrow A$, where A is an attribute. Then Σ_{FD} over D_G is defined as follows.

- For each FD $A_{i_1} \cdots A_{i_m} \rightarrow A_i \in FD$, $\{db.G.@A_{i_1}, \ldots, db.G.@A_{i_m}\} \rightarrow db.G.@A_i$ is in Σ_{FD}.
- $\{db.G.@A_1, \ldots, db.G.@A_n\} \rightarrow db.G$ is in Σ_{FD}.

The latter is included to avoid duplicates.

Example 5.3 A schema $G(A, B, C)$ can be coded by the following DTD:

```
<!ELEMENT db (G*)>
<!ELEMENT G EMPTY>
  <!ATTLIST G
            A CDATA #REQUIRED
            B CDATA #REQUIRED
            C CDATA #REQUIRED>
```

In this schema, an FD $A \rightarrow B$ is translated into $db.G.@A \rightarrow db.G.@B$. □

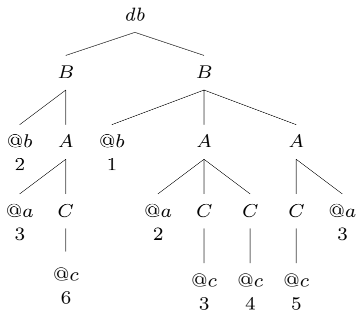
The following proposition shows that BCNF and XNF are equivalent when relational databases are appropriately coded as XML documents.

Proposition 5.4 [4] *Given a relation G and a set of functional dependencies FD over G, (G, FD) is in BCNF iff (D_G, Σ_{FD}) is in XNF.*

We conclude this section by showing that the hierarchical structure of XML documents can be used to overcome some of the limitations of relational normal forms. It is well known that every relational schema can be decomposed into an equivalent one in BCNF, but some constraints may be lost along the way. For instance, $(R(A, B, C), \{AB \rightarrow C, C \rightarrow B\})$ is a classical example of a schema that does not have any dependency preserving BCNF decomposition. By Proposition 5.4, one may be tempted to think that if we translate this schema into XML, the situation will be similar; every XNF decomposition of the translated schema should lose some constraints. The following example, taken from [18], shows that this is not the case, as we can use the hierarchical structure of XML to obtain a dependency preserving XNF decomposition.

Example 5.5 Let $D = (L_0, P, R, db)$ be a DTD, with $L_0 = \{db, A, B, C\}$, $P = \{db \rightarrow B^*,\ B \rightarrow A^*,\ A \rightarrow C^*,\ C \rightarrow \epsilon\}$, $R(db) = \emptyset$, $R(A) = \{@a\}$, $R(B) = \{@b\}$ and $R(C) = \{@c\}$. XML trees conforming to D are used to store instances of relation $R(A, B, C)$ as shown in the following figure:

A	B	C
2	1	3
2	1	4
3	1	5
3	2	6



Let Σ be the following set of XFDs over D:

$$\begin{aligned}
db.B.@b &\rightarrow db.B,\\
\{db.B,\ db.B.A.@a\} &\rightarrow db.B.A,\\
\{db.B.A,\ db.B.A.C.@c\} &\rightarrow db.B.A.C,\\
\{db.B.A.@a,\ db.B.@b\} &\rightarrow db.B.A.C.@c,\\
db.B.A.C.@c &\rightarrow db.B.@b.
\end{aligned}$$

The first three XFDs indicate how the instances of R are stored as XML trees conforming to D. Thus, for example, the first XFD indicates that all the tuples in R with the same value of attribute B are grouped together, in a tree that stores in its root the value of attribute B. The last two XFDs are the translations of relational FDs $AB \rightarrow C$ and $C \rightarrow B$, respectively.

XML specification (D, Σ) is equivalent to our original relational specification $(R(A, B, C),\ \{AB \rightarrow C,\ C \rightarrow B\})$. In fact, all the constraints in our original relational schema can be inferred from (D, Σ). Moreover, it is easy to prove that (D, Σ) is in XNF. $\square$

The approach shown in the previous example was proposed in [18], under the name of hierarchical translation of relational schemas, as a way to obtain dependency preserving decompositions of relational schemas. The advantages and limitations of this approach are explored in [18].

6 The Complexity of Testing XNF

In the previous section, we introduce XNF, a normal form for XML specifications. As in the case of relational databases, testing whether an XML schema is in XNF involves testing some conditions on the functional dependencies implied by the specification. In this section, we present some results on the complexity of the implication problem for XFDs, and then we use these results to establish some complexity bounds for the problem of testing whether an XML specification is in XNF.

Throughout the section, we assume that the DTDs are non-recursive. This can be done without any loss of generality. Notice that in a recursive DTD D, the set of all paths is infinite. However, a given set of XFDs Σ only mentions a finite number of paths, which means that it suffices to restrict one's attention to a finite number of "unfoldings" of recursive rules.

6.1 The implication problem for XFDs

Although XML FDs and relational FDs are defined similarly, in this section we show that the implication problem for the former class is far more intricate.

Regular expressions used in DTDs are typically rather simple, which allows for efficient implication algorithms. We now formulate a criterion for simplicity that corresponds to a very common practice of writing regular expressions in DTDs [6, 8]. Given an alphabet E, a regular expression over E is called *trivial* if it is of the form $s_1, \ldots, s_n$, where for each s_i there is a letter $a_i \in E$ such that s_i is either a_i or $a_i?$ or a_i^+ or a_i^*, and for $i \neq j$, $a_i \neq a_j$. We call a regular expression s *simple* if there is a trivial regular expression s' such that any word w in the language denoted by s is a permutation of a word in the language denoted by s', and vice versa. Simple regular expressions were also considered in [1] under the name of *multiplicity atoms*.

For example, $(a|b|c)^*$ is simple: a^*, b^*, c^* is trivial, and every word in $(a|b|c)^*$ is a permutation of a word in a^*, b^*, c^* and vice versa. Simple regular expressions are prevalent in DTDs [6]. We say that a DTD D is *simple* if all productions in D use only simple regular expressions. It turns out that the implication problem for simple DTDs can be solved efficiently.

Theorem 6.1 [4] *The implication problem for XFDs over simple DTDs is solvable in quadratic time.*

In a simple DTD, disjunction can appear in expressions of the form $(a|\epsilon)$ or $(a|b)^*$, but a general disjunction $(a|b)$ is not allowed. For example, the following DTD cannot be represented as a simple DTD:

```
<!ELEMENT uni (student*)>
<!ELEMENT student ((name | FL), grade)>
<!ELEMENT name (EMPTY)>
  <!ATTLIST name
            value CDATA #REQUIRED>
<!ELEMENT FL (first, last)>
<!ELEMENT first (EMPTY)>
```

```
    <!ATTLIST first
              value CDATA #REQUIRED>
  <!ELEMENT last (EMPTY)>
    <!ATTLIST last
              value CDATA #REQUIRED>
```

In this example, every student must have a name. This name can be a string or it can be a composition of a first and a last name. It is desirable to express constraints on this kind of DTDs. For instance,

$$\{\texttt{uni.student.FL.first.@value}, \texttt{uni.student.FL.last.@value}\} \rightarrow \texttt{uni.student},$$

is a functional dependency in this domain. It is also desirable to reason about these constraints. Unfortunately, allowing disjunction in DTDs, even in some restricted ways, makes the implication problem for XFDs harder.

A regular expression s over an alphabet E is a *simple disjunction* if $s = \epsilon$, $s = \ell$, where $\ell \in E$, or $s = s_1|s_2$, where s_1, s_2 are simple disjunctions over alphabets E_1, E_2 and $E_1 \cap E_2 = \emptyset$. We say that a DTD D is *disjunctive* if every production in D uses a regular expression of the form $s_1, \ldots, s_m$, where each s_i is either a simple regular expression or a simple disjunction over an alphabet E_i ($i \in [1, m]$), and $E_i \cap E_j = \emptyset$ ($i, j \in [1, m]$ and $i \neq j$). It turns out that the implication problem for disjunctive DTDs is coNP-complete.

Theorem 6.2 [4] *The implication problem for XFDs over disjunctive DTDs is coNP-complete.*

The previous result can be extended to a larger class of DTDs that was called *relational DTDs* in [4]. In some sense these results are positive, as we can expect that in practice we will be able to solve the implication problem for disjunctive and relational DTDs, since XML specifications tend to be relatively small in practice, as opposed to XML documents which can be very large, and today we can find SAT solvers like BerkMin [17] and Chaff [22] that routinely solve NP problems with thousands of variables.

We conclude this section by showing that the implication problem for XFDs is decidable. The exact complexity of this problem is unknown.

Theorem 6.3 [2] *The implication problem for XFDs over DTDs is solvable in co-NEXPTIME.*

6.2 Testing XNF

Testing whether an XML specification (D, Σ) is in XNF involves testing a condition on the functional dependencies implied by (D, Σ). Since this set of XFDs can be very large, it is desirable to find ways to reduce the number of XFDs to be considered. In the previous section we introduce the class of disjunctive DTDs. This class has the following useful property that lets us find efficient algorithms for testing XNF.

Proposition 6.4 [4] *Given a disjunctive DTD D and a set Σ of XFDs over D, (D, Σ) is in XNF iff for each nontrivial XFD $X \rightarrow p.@a \in \Sigma$, it is the case that $X \rightarrow p \in (D, \Sigma)^+$.*

From this and Theorems 6.1 and 6.2 we derive:

Corollary 6.5 [4] *Testing if (D, Σ) is in XNF can be done in cubic time for simple DTDs, and is coNP-complete for disjunctive DTDs.*

CoNP-completeness also holds for the case of relational DTDs [4]. As pointed out in the previous section, this theorem can also be seen as a positive result, as we can expect that in practice we will be able to test whether a specification is in XNF, since XML specifications tend to be relatively small in practice and today we can find SAT solvers that routinely solve NP problems with thousands of variables [17, 22].

We conclude this section by pointing out that from Theorem 6.3, we know that the problem of testing whether an XML specification is in XNF is decidable. The exact complexity of this problem is an open problem.

7 Justifying XNF

What constitutes a good database design? This question has been studied extensively, with well-known solutions presented in practically all database texts. But what is it that makes a database design good? This question is usually addressed at a much less formal level. For instance, we know that BCNF is an example of a good design, and we usually say that this is because BCNF eliminates update anomalies. Most of the time this is sufficient, given the simplicity of the relational model and our good intuition about it.

Several papers [15, 26, 21] attempted a more formal evaluation of normal forms, by relating it to the elimination of update anomalies. Another criterion is the existence of algorithms that produce good designs: for example, we know that every database scheme can be losslessly decomposed into one in BCNF, but some constraints may be lost along the way.

The previous work was specific for the relational model. As new data formats such as XML are becoming critically important, classical database theory problems have to be revisited in the new context [25, 23]. However, there is as yet no consensus on how to address the problem of well-designed data in the XML setting [12, 4, 28].

It is problematic to evaluate XML normal forms based on update anomalies; while some proposals for update languages exist [24], no XML update language has been standardized. Likewise, using the existence of good decomposition algorithms as a criterion is problematic: for example, to formulate losslessness, one needs to fix a small set of operations in some language, that would play the same role for XML as relational algebra for relations.

This suggests that one needs a different approach to the justification of normal forms and good designs. Such an approach must be applicable to new data models *before* the issues of query/update/constraint languages for them are completely understood and resolved. Therefore, such an approach must be based on some intrinsic characteristics of the data, as opposed to query/update languages for a particular data model. Such an approach was introduced in [5] based on information-theoretic concepts, more specifically, on measuring the information content of the data (or entropy [11]) of a suitably chosen probability distribution. The goal there was twofold. It was introduced an information-theoretic measure of "goodness" of a design, and tested in the relational world. It was shown in [5] that the measure can be used to characterize familiar relational normal forms such as BCNF, 4NF, and PJ/NF. Then the measure was applied in the XML context to show that it justifies the normal form XNF.

To present the information-theoretic measure proposed in [5], we need to introduce some terminology. Let $D = (L_0, P, R, r)$ be a DTD, Σ a set of constraints over D and $T = (N, E)$ an XML tree conforming to D and satisfying Σ. Then the set of positions in T, denoted by $Pos(T)$, is defined as $\{(x, @a) \mid x \in N,\ @a \in R(\lambda(x))\}$, and the active domain of T is defined as the set of all values of attributes in T. Without loss of generality, we assume that the active domain of every XML tree is contained in $\mathbb{N}^+$.

The goal in [5] is to define a function $\text{INF}_T(p \mid \Sigma)$, the information content of a position $p \in Pos(T)$ with respect to the set of constraints Σ. To do this, it is defined, for each $k > 0$, a function $\text{INF}^k_T(p \mid \Sigma)$ that would only apply to instances whose active domain is contained in $\{1, \ldots, k\}$. More precisely, let $X \subseteq Pos(T) - \{p\}$. Suppose the values in those positions X are lost, and then someone restores them from the set $1, \ldots, k$. It is measured how much information about the value in p this gives by calculating the entropy [11] of a suitably chosen probability distribution[1]. Then $\text{INF}^k_T(p \mid \Sigma)$ is defined as the average such entropy over all sets $X \subseteq Pos(T) - \{p\}$, which corresponds to a conditional entropy. The ratio $\text{INF}^k_T(p \mid \Sigma)/\log k$ is used in [5] to tell how close the given position p is to having the maximum possible information content, for XML trees with active domain in $[1, k]$ (recall that the maximum value of entropy is $\log k$ for a discrete distribution over k elements). Then measure $\text{INF}_T(p \mid \Sigma)$ is taken to be the limit of these ratios as k goes to infinity. It is shown in [5] that such a limit exists for every XML tree and set Σ of XFDs.

$\text{INF}_T(p \mid \Sigma)$ measures how much information is contained in position p, and $0 \leq \text{INF}_T(p \mid \Sigma) \leq 1$. A well-designed schema should not have an instance with a position that has less than maximum information, as we do not expect to have redundant information on any instance of this schema. This motivates the following definition:

Definition 7.1 [5] *An XML specification (D, Σ) is* well-designed *if for every XML tree conforming to D and satisfying Σ, and every $p \in Pos(T)$, $\text{INF}_T(p \mid \Sigma) = 1$.*

As in the case of relational databases (in particular, BCNF and 4NF), it is possible to show that well-designed XML and XNF coincide.

Theorem 7.2 [5] *Let D be a DTD and Σ a set of XFDs over D. Then (D, Σ) is well-designed if and only if (D, Σ) is in XNF.*

It is worth mentioning that an alternative notion of redundancy for XML trees is introduced in [28], where it is proved that an XML specification is in XNF if and only if no tree conforming to the specification contains redundant information.

8 Final Remarks

In this paper, we show how the concepts of database design and normal forms can be extended to XML

[1]Due to the lack of space we cannot formally introduce the probability distributions used in the definition of $\text{INF}_T(p \mid \Sigma)$. See [5] for a detail description.

7

databases. More precisely, we introduce a functional dependency language for XML, we use this language to define a normal form for XML specifications, and we justify this normal form by using an information-theoretic approach.

To conclude the paper, we provide some links for further reading. Many dependency languages have been proposed for XML, see [7] for an early survey on this subject and [3] for a more recent survey including results on the complexity of reasoning about constraints in the presence of DTDs. Some of these languages have been used to define other normal forms for XML such as X3NF [18] and 4XNF [27]. The information-theoretic approach presented in Section 7 has also been used to provide justification for "non-perfect" relational normal forms such as 3NF [19].

Acknowledgments

The research presented in this article was conducted while the author was a Ph.D student at the University of Toronto under the supervision of Professor Leonid Libkin.

References

[1] S. Abiteboul, L. Segoufin, and V. Vianu. Representing and Querying XML with Incomplete Information. *TODS*, 31(1):208–254, 2006.

[2] M. Arenas. *Design Principles for XML Data*. PhD thesis, University of Toronto, 2005.

[3] M. Arenas, W. Fan, and L. Libkin. Consistency of XML Specifications. In *Inconsistency Tolerance*, pages 15–41, 2005.

[4] M. Arenas and L. Libkin. A Normal Form for XML Documents. *TODS*, 29(1):195–232, 2004.

[5] M. Arenas and L. Libkin. An Information-Theoretic Approach to Normal Forms for Relational and XML Data. *JACM*, 52(2):246–283, 2005.

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

[7] P. Buneman, W. Fan, J. Siméon, and S. Weinstein. Constraints for Semi-structured Data and XML. *SIGMOD Record*, 30(1):47–45, 2001.

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

[9] E. F. Codd. Further Normalization of the Data Base Relational Model. In *Data base systems*, pages 33–64. Englewood Cliffs, N.J. Prentice-Hall, 1972.

[10] E. F. Codd. Recent Investigations in Relational Data Base Systems. In *IFIP Congress*, pages 1017–1021, 1974.

[11] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.

[12] D. Embley and W. Y. Mok. Developing XML Documents with Guaranteed "Good" Properties. In *ER*, pages 426–441, 2001.

[13] R. Fagin. Multivalued Dependencies and a New Normal Form for Relational Databases. *TODS*, 2(3):262–278, 1977.

[14] R. Fagin. Normal Forms and Relational Database Operators. In *SIGMOD*, pages 153–160, 1979.

[15] R. Fagin. A Normal Form for Relational Databases That Is Based on Domians and Keys. *TODS*, 6(3):387–415, 1981.

[16] W. Fan and J. Siméon. Integrity Constraints for XML. In *PODS*, pages 23–34, 2000.

[17] E. Goldberg and Y. Novikov. BerkMin: A Fast and Robust Sat-Solver. In *DATE*, pages 142–149, 2002.

[18] S. Kolahi. Dependency-Preserving Normalization of Relational and XML Data. In *DBPL*, pages 247–261, 2005.

[19] S. Kolahi and L. Libkin. On Redundancy vs Dependency Preservation in Normalization: An Information-Theoretic Study of 3NF. In *PODS*, pages 114–123, 2006.

[20] M.-L. Lee, T. W. Ling, and W. L. Low. Designing Functional Dependencies for XML. In *EDBT*, pages 124–141, 2002.

[21] M. Levene and M. Vincent. Justification for Inclusion Dependency Normal Form. *TKDE*, 12(2):281–291, 2000.

[22] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an Efficient SAT Solver. In *DAC*, pages 530–535, 2001.

[23] D. Suciu. On Database Theory and XML. *SIGMOD Record*, 30(3):39–45, 2001.

[24] I. Tatarinov, Z. Ives, A. Halevy, and D. Weld. Updating XML. In *SIGMOD*, pages 413–424, 2001.

[25] V. Vianu. A Web Odyssey: from Codd to XML. In *PODS*, pages 1–15, 2001.

[26] M. Vincent. Semantic Foundations of 4NF in Relational Database Design. *Acta Informatica*, 36(3):173–213, 1999.

[27] M. Vincent, J. Liu, and C. Liu. A Redundancy Free 4NF for XML. In *XSym*, pages 254–266, 2003.

[28] M. Vincent, J. Liu, and C. Liu. Strong Functional Dependencies and their Application to Normal Forms in XML. *TODS*, 29(3):445–462, 2004.

Gerome Miklau Speaks Out
on His SIGMOD Distinguished Dissertation Award, How Great It Is to Be a Professor, and More

by Marianne Winslett

http://www.cs.umass.edu/~miklau/

I would like to take a moment to thank the many people who have helped me devise interview questions over the years. Often people propose questions under a promise of anonymity, so I will not name the individuals who have suggested questions---but you know who you are! Without you, these columns would not be possible. Thank you for your many excellent suggestions over the years.

Welcome to this installment of ACM SIGMOD Record's series of interviews with distinguished members of the database community. I'm Marianne Winslett, and today we are at the SIGMOD 2006 conference in Chicago, Illinois. I have here with me Gerome Miklau, who is an assistant professor of Computer Science at the University of Massachusetts at Amherst. Gerome is the recipient of the 2006 ACM SIGMOD Dissertation Award, for his dissertation entitled Confidentiality and Integrity in Distributed Data Exchange*. His PhD is from the University of Washington, where his advisor was Dan Suciu. So, Gerome, welcome!*

I'm delighted to say that Gerome is the first *recipient of the SIGMOD Dissertation Award. Probably many of our readers haven't heard about this new award, which will be given every year to recognize excellent research in a database dissertation that was submitted during the previous year to a computer science department anywhere in the world. Our runners-up for the award this year were Marcelo Arenas from the Pontificia Universidad Católica de Chile, and Yanlei Diao, who is also at the University of Massachusetts at Amherst.*

So, Gerome, what is the thesis of your thesis?

My thesis is a response to the fact that, as a result of successes in the database community, there has been an explosion in the collection of data, and an explosion in the exchange and sharing of data. As a result, there are new challenges in balancing the need to share and exchange data and the need to protect sensitive data.

What do you see as the threats?

The general high level threat is that sensitive data will be misused. That misuse can cause harm, either by violating our personal privacy, by disclosing facts about us that we don't want others to know; or through incorrect data introduced into records about us, such as medical records and credit reports.

How does your thesis address those issues?

The goal of my thesis is to provide some theoretical techniques, and some practical tools. On the theoretical side, I address the challenge of understanding information disclosure in databases. The challenge here is to understand, when we decide to share data by publishing a view, what exactly that view may reveal about other facts that we would like to protect. We recognized that there was a lack of good definitions for understanding that disclosure precisely. We proposed a new definition, and analyzed it theoretically, providing complexity results and decision procedures and so forth. So that is the theoretical side of the work.

The more practical side of the work has to do with exchanging data, beyond a trusted domain. In a trusted domain, you can use your own systems to negotiate access to data. When you publish data beyond that trusted domain, you can't rely on others to respect the policies that you want respected. So, we rely on cryptography to enforce access control. We devised a framework for publishing data and passively enforcing access control with cryptography.

Is it a solution based on keys that you give to the authorized people?

Yes, that's right. The goal is to move from a model where you would publish one version of the data for each of the authorized recipients (of which there could be hundreds or thousands), and to instead construct a single version of the data, protected and partially encrypted in a somewhat complicated way, and publish that single version to a web server. The recipients can all take that data, but require appropriate cryptographic keys to process the data.

Are you using some sort of group cryptography scheme? How does the crypto part work?

It is fairly straightforward. There is some connection to secret sharing schemes in cryptography. I think the main insight is the view; we work with XML data, we view the data as a tree, and we annotate the tree with key expressions. So access to the data must be from traversing the tree, and you must satisfy the key expressions by possessing the appropriate keys. There is a precise semantics to access in this model, which makes it easy to analyze and easy to work with.

Do the keys intuitively express group membership, or some quality of the recipient, or are they related to the recipient's identity?

They are related to the recipient's identity and they are derived from the access control policy. The way you would construct this partially encrypted data is first to state an access control policy over the XML data in a declarative way. Then in the processing phase, the keys are actually generated automatically. An aspect of the keys corresponds to the identity of the recipients, and another aspect corresponds to the positions in the data that they have access to.

Does the policy list the identities of the authorized people, or their properties or characteristics?

Identities, but it could also support properties.

What do you see as the major open problems in the area where you were working?

There was one really challenging part of this problem that it was not possible for me to address in my work. The challenge was to construct a proof that the encrypted data produced by our process has had the encryption functions applied correctly and that the only information an attacker could get is the information implied by our precise access semantics, which is more at a logical level. This kind of problem has been addressed by others in other settings. Martin Abadi and Bogdan Warinschi wrote a follow-on paper that appeared in PODS last year and proved the security of our constructions. I view that as an important piece of follow-on work. It was really beyond my scope, as I am not a cryptographer.

Where do you see this work being applied?

With regard to the disclosure work, our results provide a theoretical ideal of security. We have done some follow-on work to make these notions more practical in nature. Some recent work that appeared at this conference (SIGMOD 2006) has improved our complexity bounds for special cases. That improves the practicality of our work as well.

I think that there is a great opportunity to make progress and achieve this balance of sharing data and also protecting it. We need good measures to help us understand what is disclosed when you publish an anonymized data set, and that is something that has trailed behind techniques for *producing* anonymized data sets. In addition, current techniques do not handle many common data sets. There are pressing real-world problems in this area. One example is the case of networked data sets. As you may have heard, recently the National Security Agency has collected and analyzed records of phone calls made by US residents. Apparently the NSA does not anonymize those records, but they probably should. If they did, they would have a massive graph of relationships that they would like to study for patterns. So an important open question is, can you publish a graph like that about entities and their relationships and provide a resistance to re-identification of individuals. There are some connections to the techniques of k-anonymity, but k-anonymity is intended for tables that represent entities, not relationships between entities.

You are the only assistant professor that I have ever interviewed, so you have a unique perspective to offer our readers. What do you know now that you wish you had known as a graduate student?

During my first year as an assistant professor I've confirmed that academia is great. You can never be sure until you get there! I am not sure what I would have done differently had I known that earlier, but I think I would have been able to act with more certainty. Maybe I would have finished my PhD sooner, had I known that this job would suit me to the degree it does, how exciting it is to work with students and come up with new ideas, and to teach.

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

I have a small stack of books on my desk about various aspects of privacy in society, written by non-technical people. These books talk about the legal and philosophical foundations of privacy, as well as practicalities about how data is collected and who legally owns data. I feel strongly that it would benefit my research if I had time to read more of these books. Understanding the societal background and history of privacy would provide inspiration for my research.

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

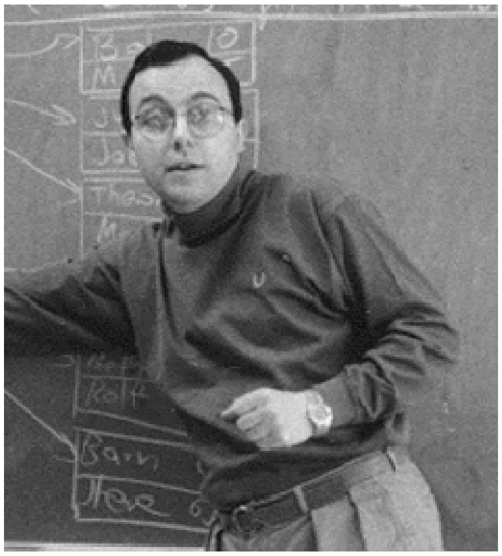
A stronger background in statistics would benefit me, I think. My background is more in logic and discrete math, and certainly there seems to be a trend towards the increased importance of statistical modeling in computer science.

Thank you very much for talking with me today.

Thank you.

Yannis Ioannidis Speaks Out
on Database Research Funding in Europe, the Importance of Being Uncertain, Teaching as Show Business, the History of Histograms, and More

by Marianne Winslett



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

Welcome to this installment of ACM SIGMOD Record's series of interviews with distinguished members of the database community. I'm Marianne Winslett, and today we are at the SIGMOD 2006 conference in Chicago. I have here with me Yannis Ioannidis, who is a professor of informatics and telecommunication at the University of Athens. Before that, he was a professor for many years at the University of Wisconsin in Madison. Yannis's research focuses on query optimization, digital libraries, and management of scientific data. Yannis is an ACM Fellow, has won numerous teaching awards, and received the VLDB 10 Year Best Paper Award. He is the Vice-Chair of SIGMOD, and his PhD is from the University of California at Berkeley. So, Yannis, welcome!

Thank you.

Yannis, your VLDB Ten Year Best Paper Award was for your 1993 paper, "Universality of Serial Histograms." Your paper was one of the first to introduce histograms in a principled way. How do people use histograms in the database world?

They use them to estimate how data values are distributed inside the database, and based on that, figure out which way to process queries. In the beginning, database systems had very rudimentary histograms, just assumptions that data values were uniformly distributed. With more sophisticated histogram, query optimizers have become much more accurate---not completely accurate in all cases, but much more accurate. And now all major commercial database systems use histograms in one form or another.

What were the research challenges in histograms for query optimization, and what influence has your paper had?

Finding what histogram to use was the biggest research problem. In the beginning, when people moved away from the uniformity assumption, they used the so-called "equi-width" histograms. And then they talked about "equi-depth" histograms, which were much better. And that was pretty much it. The main influence of my original 1993 paper was to show that there is a whole family of other histograms that will give much better results in query optimization, and are much more accurate, for many distributions of data values. And then all the other papers that followed that, both from my group or from many other researchers, really nailed the problem of what approximation is the best approximation for histograms.

Are there any new data management challenges that the database research community needs to be aware of?

The sky is the limit! There is so much that the community could be doing that it is not doing. I think we will be busy for many years to come.

One challenge is uncertainty. In the relational world where we grew up, data and queries are well structured and pretty well understood. You can do lots of things with relational data, and you know that you will get them right. Most of the world is not like that. We need to provide data management techniques that can handle uncertainty in the usage of data, in the accuracy of data, the provenance of data, and so on.

A second challenge is to serve all the other communities. There is a tremendous number of applications with huge data management problems, and we are not paying enough attention to them. I think we should. For example, we should consider how to deal with scientific data; how to deal with medical data and biological data; how to mix varied forms of data, text, video, images, and unstructured data. This is a huge challenge that we should be diving into.

Funding is a good carrot to get database researchers to go in new directions. Now that research funding is so tight for mainstream database research, I see more and more people teaming up with scientists. Certainly at every school in the US midwest, people are working with scientists and getting money through new avenues. What about the multimedia direction that you mentioned---handling different types of content?

Multimedia is at the heart of many of these applications. You cannot do many forms of science unless you have images from satellites. Much of medicine involves imaging and clinical data. All this multimedia comes to your table to be studied with all these challenges. As long as you try to serve other communities, you will hit the multimedia issues right away.

How hard was it to leave your flourishing career in the US and start over from scratch?

It was very hard. I was a member of what was considered the number one database group at the time, in Wisconsin. It was very hard to say yes to my family and to myself as well, and go back to Greece. And, you said it exactly right, it was pretty much starting from scratch. The system is very different, you have to establish new labs, and get new people. Also, the funding schemes are different, and so you have to start thinking in a different way; you have to change problems, because different things are funded on the two sides of the ocean. So, it was hard, but worth it in the end.

Can you comment on how the funding is different on the two sides of the ocean for database research?

Yes, and I will try to be politically correct in my answer. In the US, if you have a bright idea, as an individual, you can go get a small grant and study your idea and see how far you can go. There is not very much money, but it is there for you to do this. In Europe, there is tons of money, but there is no avenue to explore basic very long term ideas. You have to team up with lots of other companies, universities, and institutions from many countries, have a grandiose scheme, which must have some plan even for commercial exploitation, and that is how you get money. And the basic research, to a large extent, is done under the covers. So that is the main difference. Also, on a second level, because of that, the kinds of problems that get funding are different. There is no database research funded in Europe. Maybe this is because the main companies for database technology, for database servers, are in the US. So in Europe, officially, I don't do database research, I do other things. And, of course, through my funding that I get for other things, I do also my database research!

So what kind of other things did you migrate to?

Digital libraries, for example. And I don't complain about that, it is a fascinating field. I get inspired a lot by the problems that I meet there, both to do digital library research per se, and to change my database research. I am also involved now in some e-health projects where I am trying to apply data management techniques to health systems. There's no right or wrong funding system, it is just different in different places. In moving to Greece, and to Europe in general, I had to change my focus.

Do you find that you spend your time differently now that you are a professor in Greece, versus how you spent your time in the US?

Yes, because of the different ways things work in Europe. I spend more time connecting with people, both to come up with ideas to write proposals, and also to do the work. The projects in Europe are a lot more administration-heavy, so I have to deal with that overhead. Teaching load is heavier, in Greece at least. I teach two courses every semester. Classes are much larger. The courses that I used to teach would have 60 undergraduate students in Madison, and in Greece I have 200 students. In the graduate courses in Madison I used to have 20 or 30 students, and in Greece I have 80 students. So, all of this has an effect to have less time to spend on research, or less personal time. Another thing I am trying to spend some time on in Greece is trying to change some things that I saw in the US and I liked. Together with some of my colleagues, we are trying to change people's attitude towards certain issues. So I am putting some time into, let's say, foundational changes---sometimes successfully, sometimes not successfully, but it's something I am doing.

There seems to be a bit of a reverse migration of database researchers back to their home countries in Europe. Have conditions changed in some way that makes this a more attractive route than it was before?

Absolutely. I believe the European Union has played a major role there. In the past 20 years, the EU started funding "research framework" programs. Technological areas, computer science in particular, have a constant stream of money going through these programs. So the ability to do research, or at least the funding for research and development in an academic environment, has increased and keeps increasing, and there is no sign of it diminishing. Since the opportunities are there, it is much easier for someone like me to return, and with some effort, to rebuild and do partly, at least, what he or she was doing back in the US.

What do you think of "publish-or-perish"?

I believe it is wrong, when it comes to the quantity of publishing. Obviously you have to publish, nobody would remember you for ideas that you never spoke of or never wrote, so you have to publish. But, I would rather see someone with three good papers than someone with one good paper and 10 mediocre papers. Often people are judged first by the numbers, and then by the quality. Not always, but often. I would say it is better to spend more time on a problem and do it right, and have one really good paper, rather than the other way around.

I hear that you are a fantastic father. How do you reconcile the time commitments for a good family life and a good research career?

I am sure you haven't heard that from my children! I don't consider myself a fantastic father. I think I could be much better than I am. It is very hard for anyone to reconcile those time commitments. I'm trying to spend as much time with my children as possible. At least, if I don't spend as much time with them as I would like, I want the times that I am with them to be good times, educational times, and fun times. It is often hard now that I am in Europe and all these research projects have so many partners at remote sites. Plus I travel quite a bit more than I used to before, say 10 years ago. I am often away, and that is really tough on me and my kids as well. I used to buy tickets so that I would have a Saturday night stayover, I don't do that any more!

I have heard that you are a very hands-on advisor. With that advising style, how can the students be ready to "fly solo" when they leave your nest? And more generally, what have you found to work well in advising students?

I think it is accurate that I am "hands-on", but not in the sense that I hold them by the hands and guide them. I am just there next to them. In the beginning, I hold their hands, and then, little by little I let them go. But I am just there, I want to understand the details in order to teach them about the details when I see something wrong. So, in the end, I believe they will end up being stronger and readier to fly solo than if I had just thrown them in deep water and asked them to swim. I consider this a benefit as opposed to a disadvantage.

Being a friend in addition to an advisor has worked well for me in advising. Getting a PhD is an issue of self-discipline and growing as a person, as much as it is growing as a scientist. It is a lonely way with some ups and many downs. Until you get your first SIGMOD or VLDB paper, you have, more often than not, rejections in other places. Students can lose courage when their first attempt at a paper is not successful. So you have to be there to support them, to teach them patience. I think this is the most critical thing, next to the scientific things that you teach them; to be there with them and support them psychologically.

I have heard that you enjoy being on stage, and were involved in the theater in Greece as a youngster. How does that affect your teaching and speaking style?

Very much so. I think the way I teach is a 100% influence of my theatrical experience when I was in high school. When you are up on stage, you have to communicate with the audience, forget who you are, immerse yourself into your role, and then give this to the audience. I think that is the way I teach. I forget who I am and I immerse myself into the topic that I am supposed to teach, and I try to communicate that. However, in the theater, there is not much interaction with the audience, other than that you are looking at them. In some modern theater you may have more interaction, but in classical theater you don't. When teaching, you have the additional advantage of being able to explicitly interact with the audience, and then sense how things are going, and proceed accordingly to convey that. I think teaching is show business, just like theater. I became a different person after my theatrical experience than before.

You were an advisor to the Minister of Health in Greece. What was that like?

A very different experience. I was called to be an advisor because of my expertise on data management and computer science in general. On 5% of the cases, I used that expertise, and then the remaining 95% was completely new things: how to manage large organizations like the ministry, how to say what you need to say without offending people because of politics. There was a lot of public relations work, too---not that I did all this, but I had to face all this, so it was a great learning experience. I don't know if I would ever do that again, but I learned a lot.

Often, as researchers, we have the greatest ideas and we come up with the right solutions, and then we try to apply them to reality. But then the human factor comes in, and issues that you never thought of are raised as problems, and resistance arises to applying these wonderful new things that could change the world. You don't know what to do, you speak a very different language from the rest of the people. Often I felt that living in academia or in computer science in general, we are in our own little world. We think that the rest of the world operates like us, but it does not. It is a wild world out there! It was fun, but I hope it was the first and the last time I did that.

Your interests in scientific data management have led you to work with real scientists. How was that experience?

Very interesting. It is trying to interact with a different world. It takes a long time for the two communities, computer scientists and people from other sciences, to understand each other, to start speaking the same language. So it is a lot of effort up front, but it is very rewarding. In high school and somewhat in college, you learn about rudimentary things from other sciences, and then you forget about them because you dive into your own science. Working with scientists gives the opportunity to bring back all the subjects that you used to like back in high school - for me biology, astronomy, physics, and other things. It gives the opportunity to learn some more about them, and also, to a certain extent, see what we do in computer science be useful. Often when you work in the internals of servers you never see the results of that work affecting people. A few cases where I saw that were extremely satisfying. I recommend it to all of us.

At Wisconsin, the computer science department gives out a teaching award, and the students also choose a recipient of a teaching award. You won both awards so many times that I heard that they made a special award just for you. I have heard that you are especially good at teaching database theory to systems students. How do you do that?

Indeed there was a special award just for me when I was leaving Wisconsin. They gave me a lifetime award, which was really nice of them. It is very hard to say what it is that I do that makes my teaching style successful. Here are a couple of things that I can think of. The first is the interactive style that I mentioned before. Often I stop, I ask questions, I ask people to vote on ideas, I ask them to give me answers, then the class votes on those answers, and this keeps a good percentage of the audience engaged. I think this plays a major role. The other thing is that often, I don't just teach what it is, but I try to bring it out from them. I lead them along the way, I ask them questions, and wait for responses from them. And if they say it, or one of them says it in a way that they all understand, it stays in their minds.

So how do you do that now that your class size is 300 instead of 30?

The same old way. I don't use PowerPoint. In undergraduate courses, I still teach on the board. The pace is slower so people can follow more easily. I often try to lead the class into a wrong answer to a question, and then bring them back and tell them what the right thing is, and I ask them not to remove it from their notes. I want them to have gone through the wrong, and then gone back to the right answer.

Don't the students complain vigorously if you don't give them PowerPoint?

No, not really. I haven't heard that complaint. Actually, I teach a human-computer interaction course in Greece, I didn't use to teach that in Madison. The past couple of years, it has been the only course that I started teaching with PowerPoint, because the topic is user interfaces, so it is much easier if the students can see the interfaces. And a few people complained. Okay, they said, use PowerPoint for the interfaces themselves, but do rest on the board.

At Illinois, if you put anything on the board, the students complain that they can't see it, because the class is so large that they can't really see the board. PowerPoint is projected very large, so everybody can see what you are doing.

In the large classes in Athens, we use a classroom where we videotape the lecture. We videotape the board, and these videos are projected on screens for the students to see. And also, you can watch from your house over webcast.

David DeWitt says that we need a new paradigm for query optimization. Do you agree? What should that paradigm look like?

I don't know if we need a new paradigm, but certainly there are lots of things we haven't solved, and we need to study them. If the same paradigm works, that is fine; otherwise we need a new one. The key thing is uncertainty: mixing optimization and query processing; optimizing for different kinds of answers, not accurate answers; optimizing not for speed but for completeness or for freshness of data; optimizing in a distributed system when you have autonomous systems that can give you some answers but you cannot control them. If I wanted to put these kinds of different environments under an umbrella, I would call them uncertain environments. They are certainly a big challenge and definitely may well need a different paradigm than normal.

You are interested in human-computer interaction (HCI) issues, which the database community has traditionally been very poor at addressing. Is there new hope for us?

I think there is hope for us if we put our minds to it. The HCI community has made tremendous progress in general with user interfaces. Database querying and database interaction are a special form of user interfaces, and may have particular difficulties that have caused us to make little progress. We can never be sure, because really we haven't spent the time necessary to know. For easy queries, we did QBE, we did query by forms, and now the web works like that. But how do you express an aggregate, how do you express nested queries? It is tough, but we haven't spent enough time working on it in order to crack it.

Should database systems people care about schema equivalence?

Stonebraker says no, only 5% of the problem is schema equivalence. I would say yes, but in a broader sense, not purely schema to schema. Most of the world is not schema based, but still you need to have mappings. Let us put all the items of the entities (those that are in the data, those that are in the schema, and anything else) into one big bag, and then see how to find equivalence across the bags.

You did research on schema equivalence in the context of the extended relational data model. What new challenges arise when we want to reason about XML schemas?

I don't think any new brand new challenges arise. It is just that the problems are harder because in the relational world you have more structure. In the XML world, you lose some of that, and then how do you map across? Some people have done some really nice work on that.

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

Think out of the box. Address challenges that nobody else is looking at. And, especially, what I said before, find challenges that other scientists, or other parts of society, have. Try to address those problems. There are many problems to be solved there. The impact would be larger in the end. I mentioned a few such areas earlier, let me not repeat them here---but go out and collaborate with other sciences. Some of us must still work on server stuff and classical problems that reoccur in new environments. But the great majority, especially fledgling and midcareer people, should go further away and dig there. There is a lot of gold there.

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

Let's see, which child of mine do I love the most?! Well, certainly, the histogram work has a special place in my heart. It was exciting and very different. Also, it has had the biggest impact of all the work I've done. But even before this was realized, it was a problem that people could have looked at many years before I did, but no one had. It was well defined, and it had a very interesting solution. So I love that work. Also, I like my work on scientific databases. It hasn't had as much impact, but I believe it is very important in the systems that my team and my colleagues build now based on the results that have come out of that. I am also excited about personalization, which is my latest project.

What challenges do you see in developing user preference models?

User preference models and personalization in general is a field where you have to move away from the machine and bring the human into the picture. And then everything breaks loose. Humans are unpredictable. We don't know exactly how humans operate. So coming up with models of preferences, how people think about preferences, is extremely challenging. Yes, I believe there is hope. There are some models, or some partial models, that seem to work well. Some of them are useful in databases and data management in general. Some others are for other types of applications. It requires, at some level at least, interaction between computer science and psychology, which makes it very exciting.

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

I would probably work on user interfaces for databases. I think there are things that can be done there. There are many other things I would like to do at work, but I think that one would be at the top.

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

For some things, I wish I knew more mathematics. It would help push certain areas deeper than I have been able to push them. And speaking of pushing, one thing that I would change in myself is the following. When you work on a problem, you work on it for a little while, and then you move on to the next problem. In certain cases, I have moved to the next problem too soon. I believe I should have stayed and pushed a bit more.

In hindsight, where would you have stayed longer?

I would stay longer on heterogeneity and schema equivalence, things like this. I would have stayed on user interfaces, I've done a little bit of work, but I haven't stayed long enough.

Thank you very much for talking with us today.

Thank you.

Changes to the *TODS* Editorial Board

Richard Snodgrass

`rts@cs.arizona.edu`

Mary Fernandez, Raghu Ramakrishnan, and Jennifer Widom rotated off the Editorial Board last month. All have been active on the editorial board, handling dozens of submissions and participating in policy discussions. In particular, Jennifer was a prime architect of *TODS'* prior publication policy (referred to affectionately as P^3), which has had a salutary effect on papers and on the reviewing process.

We as a community are extremely fortunate that scholars of the stature of Mary, Raghu, and Jennifer are willing to devote hours every week to shepherd submissions through the reviewing cycle thoroughly yet quickly. The Associate Editors, along with the reviewers and authors they work with, are the primary reason that *TODS* is so prominent.

Thank you, Mary, Raghu, and Jennifer, for your selfless, highly capable work over the last few years.

In other news, the December 2006 issue should be out by the time you read this. This is the much-anticipated SIGMOD/PODS special issue, along with several other regular papers, nine papers in all, another mega-issue.

This issue rounds out the most prolific year in *TODS* history, with 36 papers in all, five more than the largest of the past thirty volumes. Enjoy!