SIGMOD Record – June 2006 – Volume 35 Issue 2 – www.sigmod.org/record

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

SIGMOD Officers, Committees, and Awardees

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Award Recipients:

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

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

Editor's Notes

I want to make these notes short, but there is one thing which I want to address upfront. I want to apologize to authors who feel that it is taking longer to have their papers reviewed than they (and I) would like. I have received more submissions that I would have expected (which is good news) and with the conferences review cycle being practically being tied back-to-back, it has been increasingly hard to find reviewers for the submissions. On top of that, given that conferences have a harder deadline, often the reviews for non-conference publications (*SIGMOD Record* being only one of them) are often relegated to a second plan (I cannot deny that I have done it myself!). Thus, I have to ask: if you are submitting a paper, be patient; I can assure you that every paper is given proper attention, but these are not instantaneous, unfortunately. Sometimes I do take a few days to acknowledge a submission but that is just because, as most of you, I also have a day job, which is not being the *Record*'s Editor. Bottomline, please be patient when submitting papers, and, if you agree to review a paper, please do so in a timely manner.

Another topic that I want to touch upon now, even though I am not ready to establish a "policy" yet, is about the type of papers that can and should be accepted for publication in the Record. As I have discussed before (see my notes for the Dec./2005 issue) I am trying to prioritize papers that are of general interest to SIGMOD's community. On the one hand that is not to say that more technical papers should not be submitted or will not be accepted. On the other hand it is to say that if the reviewers or I feel that such technical contribution is too narrow and the submission is better suited to a specialized conference or workshop or journal, then it may not be accepted. In other words, acceptance of submissions has been and will probably become more dependent on the value of the contribution to the SIGMOD community at large than to its research-oriented value. As said, this is not a clear policy yet, but I hope to be able to soon discuss this with the Associate Editors and come up with more clear guidelines.

Besides the usual section of research papers and edited columns, this issue has two invited papers that, in my opinion, are of general interest to the *Record*'s readership, and I would to call your attention to. The first one, by Samuel Madden and David DeWitt discusses the impact of double-blind reviewing. In a day and age where the number of papers submitted to conferences are getting quickly very large I think that this kind of analysis may spark some interesting thoughts. The other paper, written by Thomas Haigh, presents, in a historical perspective, the development of our research domain. The paper is a revision of a paper that was originally printed in the Proceedings of the 2002 Conference on Heritage of Scientific and Technological Information Systems. As noted by Rick Snodgrass (who aptly suggested the revised version to be published here), SIGMOD Record is probably the ideal venue for this type of paper. I hope that you all agree with that, and enjoy this issue in its entirety.

Mario Nascimento, Editor.
May 2006.

Chair's Message

Greetings. Since I last wrote to you in this forum, I attended the meeting of the ACM SIG Governing Board (SGB) in Chicago, and we (the EC) have been discussing ways to mount a membership drive. We've been observing the enormous effort involved in organizing SIGMOD 2006, and assisting when we can. Meantime, Beng Chin Ooi has been assembling an impressive PC for SIGMOD 2007.

An important development is the new SIGMOD awards committee structure. We adopted the structure of the ACM awards committee, with five members who serve staggered terms (to ensure continuity as members rotate off the committee). I am very grateful to the following members of our community for agreeing to be on this important committee:

- Serge Abiteboul (Chair), Mike Carey, David Maier, Moshe Y. Vardi, Gerhard Weikum

Separately, we have for the first time an award for the best dissertation in the database area! Many thanks to Hans Schek and the SIGMOD Dissertation Award committee for all the effort they've put into this selecting the winner of this important new award for 2006, to be presented at SIGMOD in Chicago:

- Hans-Joerg Schek (Chair), Ricardo Baeza-Yates, Sophie Cluet, Jim Gray, Masaru Kitsuregawa, Gail Mitchell, Beng Chin Ooi

I'd like to give a special thanks to Shahram Ghandeharizadeh for his yeoman service as the editor of the DiSC anthology. He will step down shortly, and Joachim Hammer has agreed to take on this very challenging and vital role. If you see either of them, buy them coffee—hey, buy them lunch. They've earned it (or will, in the case of Joachim!).

On a personal note, after nearly 20 years at Wisconsin, I will be joining Yahoo! Research this summer. While I am excited about the new opportunities and challenges, it is with mixed emotions that I leave Wisconsin, which has been a wonderful place to live and to grow, personally and professionally.

Sincerely,
Raghu Ramakrishnan

Extending Object Database Interfaces with Fuzziness through Aspect–Oriented Design

Miguel-Ángel Sicilia & Elena García-Barriocanal
Computer Science Dept., University of Alcalá {msicilia,elena.garciab}@uah.es

Abstract

Fuzzy logic has been used yet for extending database models to deal with vagueness in the definitions of linguistic concepts as "tall" or "long". However, the extension of existing programming interfaces for fuzziness requires a proper modularization of the underlying concerns of numerical imprecision handling. Such modularization should not interfere with existing programming practices, and they should not obscure the original design. Aspect–oriented design (AOD) enables such form of non–intrusive extensions to be added to existing software libraries. In this paper, the main design and implementation issues of such AOD–based extensions on `OJB` database libraries are briefly sketched.

1 Introduction

Fuzzy set theory provides numerical models for handling vagueness as expressed for example in sentences like "x is *tall*". In such sentences, the determination of which individuals belong to the class or set *tall* is a matter of (numerical) degree that depends on their height, as modeled in what is commonly known as "membership functions". In addition, other related mathematical frameworks have provided numerical models for other kinds of imprecision or uncertainty [13]. The integration of such models into database structures has been subject of a significant amount of research in recent years. Fuzzy databases and fuzzy querying research has resulted in diverse extensions to the relational and object data models [2, 3, 5]. Further, several implementations of fuzzy queries on top of commercial database systems or standard interfaces have been developed, e.g. [9, 17, 18].

In some modern database approaches, interfaces for *orthogonal persistence* of objects [1] are provided to the programmers, i.e. the provision of persistence is the same for all data irrespective of their type. This entails that no specific programming is required to make a type persistent. The extension for fuzziness of such kind of interfaces can be accomplished by adding elements to the query syntax and also by augmenting programming interfaces to deal with the desired fuzzy capabilities — e.g. as in [9]. In any case, several reasons point to some implementation characteristics that are required for a seamless integration with existing interfaces. On the one hand, extensions should be *strictly additive*, i.e. they should not interfere with the non–fuzzy capabilities of the programming and querying interfaces, both for the sake of backward–compatibility and of ease of learning [12]. And on the other hand, the extensions should be properly *modularized*, not obscuring the original design and architecture of the extended database libraries. The first requirement can be met through a careful design, using polymorphism and proxy objects as described in [9], and using reflective capabilities when required. Nonetheless, the second requirement calls for specialized design and implementation capabilities that allow the extension of software with cross–cutting concerns (as fuzziness can be considered with regards to data representation). Aspect–oriented design (AOD) [16] provides the required modularization capabilities for the latter issue. In this paper, the implementation issues of fuzzy extensions to object–database interfaces are approached from the perspective of AOD. Such novel approach is then put into practice through a concrete case study using `aspectj` to extend the Java–based interfaces of the `ObjectRelationalBridge (OJB)`[1] open–source libraries. Related work includes studies of fuzziness related to prototype theory of categorization [15] to classify types of software units, but no previous research exists on the introduction of fuzziness in software units through aspect–oriented techniques.

The rest of this paper is structured as follows. Section 2 describes the rationale behind using AOD to implement fuzziness as a concern in existing programming interfaces. In Section 3, the general issues of extending `OJB` interfaces with fuzziness are described. Then, a simple case of fuzzy extension is sketched in

[1] http://db.apache.org/ojb/

Section 4.

2 Fuzziness as a Concern in Object-Oriented Software

Imperfection in information should be addressed early in the lifecycle due to the specifics of uncertainty and imprecision in conceptual modeling [4], and its impact on architectural and implementation decisions — most notably in persistence and querying [9]. Information imperfection is a logical "matter of interest", according to COSMOS [14] terminology. It can be organized in several *classifications* [14], one for each of the considered principal aspects of imperfection, e.g. `Imprecision-Related`, `Uncertainty-Related`, `Inconsistency-Related` or `Hybrid`. Further subdivisions inside those categories may refer to more specific types of imperfection according to a given list like Smet's taxonomy [13]. For example, `FuzzyElement` refers to "imprecision without error" without decidability as in "age is close to 30", while `PossibleElement` refers to "happen–ability" as a kind of uncertainty. According to the kind of element in the conceptual model that is subject to imperfection, we can have additional classifications in another dimension, namely `Imperfect-Element`, and `ImperfectRelationship`, the former containing subdivisions as `Imperfect-Class`, `Imperfect-Attribute`, `Imperfect-Function` and `Imperfect-Result`, which roughly correspond to classes, attributes, methods and method results in object–oriented conceptual models. Imperfect conceptual model elements can be expressed in the domain model through extensions to the *Unified Modeling Language* (UML) notation like the one sketched in [8].

AOD can be used as the candidate detailed design technique whenever concerns cross–cut software modules, and it becomes the required option if existing libraries are required to be extended for fuzziness without changing the syntax and semantics of their interfaces. This is because *aspects* like those of `aspectj`[6] allow the proper modularization and encapsulation of concerns, avoiding tangling existing source code. Moreover, AOD enhances maintainability, since the added concerns for fuzziness are separated from "crisp" functionality, so that defects can be easily located. In addition, AOD for the implementation of fuzziness eases the production of "crisp" and "fuzzy" versions of the same software, since the features of *advice* and *introduction* contained in aspects are dynamic, and thus they can be "disabled" by simply excluding the aspects from the concrete build of the system.

3 Extending `OJB` with `aspectj`

As mentioned above, the techniques of aspect-oriented design (AOD) provide improved modularity to software systems by focusing on separation of concerns [16]. Fuzziness can be considered a cross–cutting concern for existing database processing libraries, so that it can be added to existing crisp software without altering the original programming interfaces. As a proof of concept for introducing fuzziness, the `OJB` framework has been extended by using `AspectJ`. The `AspectJ` framework [6] is an AOD extension to the Java programming language based on the concept of dynamic *pointcut* (the intersection of a number of well–defined execution points) and *advice* (code attached to specific pointcuts). Using aspects in `OJB` requires a previous recompilation of its source code version. OJB supports multiple persistence application programming interfaces (APIs), including JDO and ODMG compliant ones. However, all of them are built on top of a persistence kernel, so that it makes sense to concentrate first on it, and later address higher–level interfaces. In addition, a principle of "minimum difference" with existing interfaces and programming idioms is followed, in an attempt to maximize the *usability* of the extensions in the sense described in [12].

3.1 Extending metadata handling

Metadata handling in `OJB` is centralized in the `MetadataManager` class, implementing a *singleton* pattern that can be used to obtain a reference to the `DescriptorRepository` instance containing object mapping and manipulation information for persistent objects. The class needs to be extended to deal with the required metadata describing fuzzy constructs. The better way to do it is by merging standard metadata descriptions with fuzzy ones. This can be accomplished by invoking the `mergeDescriptorRepository` method after a (successful) call to the `MetadataManager.init` method at construction time, by capturing its *pointcut* through an around *advice*. The following code fragment sketches an aspect encapsulating such processing:

```
public aspect FuzzyMetadataManagement {
  DescriptorRepository drp;
  void around (MetadataManager m):
      target(m) && call(* MetadataManager.init(..)){
```

```
        try{ proceed(m);
        }catch(MetadataException e){throw e;}
        drp = loadFuzzyDescRepository();
        m.mergeDescRepository(drp);
}
private DescriptorRepository
          loadFuzzyDescRepository(){
        DescriptorRepository dr =
                 new DescriptorRepository();
        //  load descriptor repository..
        return dr;
}    //...}
```

The `loadFuzzyDescRepository` method simply carries out the processing of the fuzzy schema description residing in a XML file, similar to that described in [9]. It should be noted that neither inheritance nor reflection would have been enough to make this change without modifying existing libraries, since at least the `getInstance` method of `MetadataManager` would had to be changed to instantiate a new subclass or build it dynamically.

3.2 Describing `Imperfect-Elements`

The current interface of `DescriptorRepository` makes use of `ClassDescriptor` for the object–relational mapping descriptions, which in turn uses `FieldDescriptor` to specify the storage of class attributes. Both elements can be extended to their fuzzy counterparts through subclassing, and other conceptual data elements like associations [10] can be derived from the common super–class `DescriptorBase` which is an open–ended hook for them. `Imperfect-Class` and `Imperfect-Attribute` concerns can be implemented by extending the `ClassDescriptor` and `FieldDescriptor` classes respectively with methods to query for the data mapping of the membership degrees, required to implement querying and storage functionality. Figure 1 depicts some of the details of such extension as a `UML` diagram. As showed in Figure 1, the `AttributeDescriptorBase` is used as an intermediate extension point for persistence mappings that are not necessarily relational, while `FieldDescriptor` provides the specific details of the relational mapping. `FuzzyFieldDescriptor` is provided as the base class for any relational–mapping of imperfect attributes, and `FuzzyNumericFieldDescriptor` is one of its subclasses representing the mapping for (triangular) fuzzy numbers. `FuzzyClassDescriptor` encapsulates the details for the relational mapping of fuzzy classes and sub–classes, with the possible variants described in [9]. *Extensional* mappings store

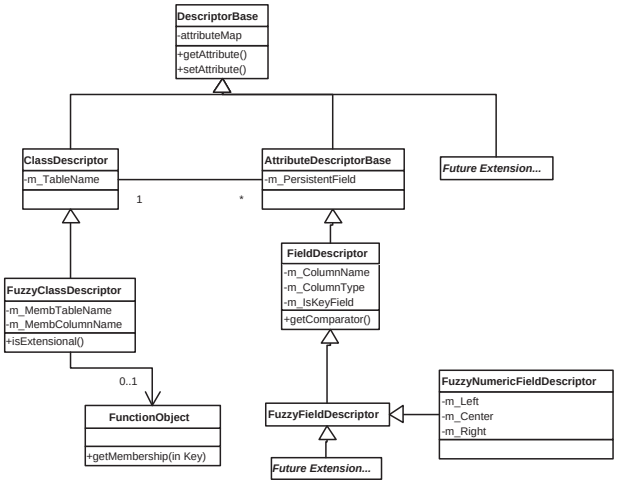


Figure 1: Extension of `OJB` description classes

explicitly the membership degrees for each object in the database, while *intensional* mappings provide a `FunctionObject` instance encapsulating arbitrarily complex computations of membership degrees for the class. In this latter case, Java's reflection capabilities enable the specification of a `FuzzyObject` subclass in the configuration file that is instantiated dynamically at run–time.

3.3 Basic fuzzy storage

Object storage both in `ODMG` and `JDO` mappings proceeds in cascade, storing the graph of references starting from the object being stored. For example, JDO provides a method `makePersistent` in the `PersistentManager` interface to make concrete instances persistent, and it also provides persistence by "reachability", so that any instance linked to a persistent one (transitively) is also made persistent. The underlying core `OJB` API ultimately uses the `store` methods in the `PersistenceBroker` interface to resolve those calls, which explicitly handles also the storage of `Collection` implementing classes. The storage of fuzzy classes only requires code modifications for *extensional* fuzzy classes, in which membership degrees are explicitly stored. To do so, variants of the `store` methods with an additional parameter are required. `Aspectj` *introductions* can be used to accomplish such extension, avoiding subclassing such a complex class like `PersistenceBrokerImpl`. The aspect can be targeted to the interface `PersistenceBroker` to guarantee that future implementation classes also are provided with fuzzy storage methods. The following code fragment sketches such extension:

```
public aspect FuzzyStorageHandler {
```

```
public void PersistenceBroker.store(
   Object obj, Double m,String fuzzyClass)
            throws PersistenceBrokerException;
public void PersistenceBrokerImpl.store(
    Object obj, Double m,String fuzzyClass)
    throws PersistenceBrokerException {
    store(obj); storeMembership(obj,m,fuzzyClass);
}
public void
    PersistenceBrokerImpl.storeMembership(
         Object obj, Double m, String fc)
         throws PersistenceBrokerException {
 FuzzyClassDescriptor c = (FuzzyClassDescriptor)
  descriptorRepository.getDescriptorFor(fc);
 if (c.isExtensional()){
   // call access layer to store m...
}  } }
```

The `fuzzyClass` attribute is required since a single object may belong to more than one fuzzy class with different degrees, i.e. multiple classification outside of the capabilities of the programming language is assumed. The storage of fuzzy attributes is delegated in `PersistenceBrokerImpl` to a `JdbcAccess` interface which acts as a layer encapsulating the construction of SQL sentences from metadata descriptors and actual objects to be stored. The delegation chain can be followed through the `StatementManager` — responsible for the value `bind`ing process — class to the `StatementsForClass` interface and implementation – which only provides caching for statement templates — and to the `SqlGenerator` interface and implementation, which actually build the SQL queries from class descriptions by delegating to a number of classes, one for each type of SQL sentence. In consequence, it is in the `SqlUpdateStatement` where the actual sentence creation logic resides, where it can be seen that no modification is required to store fuzzy classes, since it relies in the (extended through polymorphism) behavior of `ClassDescription`.

In contrast, the storage of fuzzy fields as those described by `FuzzyNumericFieldDescriptor` require dynamic extension of the methods `appendListOfColumns` and `appendListOfValues` since they assume a single table column for each field. This can be accomplished by an aspect design as the one sketched in what follows:

```
public aspect FuzzySqlFieldHandling{
  List around (SqlInsertStatement i):
    target(i) && args(cld) && args(buf)
    && call(List SqlInsertStatement.
       appendListOfColumns(
       ClassDescriptor cld, StringBuffer buf)){
     List aux=null;
     try{  aux =  proceed(m, cld, buf);
     }catch(Exception e){throw e;}
     if (cld.getClass().equals( new
        FuzzyNumericFieldDesc().getClass())){
        // add columns to SQL INSERT in buf...
     }   }  // ...}
```

The *around* advice is able to dynamically extend the behavior of the SQL–forming methods by manipulating parameters and return values.

3.4 Fuzzy querying through aspects

The abstract `SqlQueryStatement` class and its concrete subclass `SqlSelectStatement` together implement the generation of SQL SELECT clauses from queries. Queries for fuzzy classes and fuzzy attributes can be formed by extending `getStatement` invocations by using an *around* advice and an implementation technique similar to the one described above for insertions. In addition, it is required that the membership degrees of the result collections are differentiated from standard attributes of the objects. To do so, query results returning from methods like `getCollectionByQuery` in `PersistenceBroker` need to be wrapped into objects representing pairs $(o, \mu_q(o))$. This must be done at the level of the implementation of the method `getCollectionByQuery` in `PersistenceBrokerImpl` since memberships come from the access layer as conventional retrieved database columns.

```
public aspect FuzzyObjectWrapping{
    Collection around (PersistenceBroker p):
        target(p) && args(cld) && args(buf)
        && call(Collection
     PersistenceBroker.getCollectionByQuery(
          Query query)){
           Collection aux=null;
          try{
              aux =  proceed(p, query);
          }catch(PersistenceBrokerException e){throw e;}
        return this.wrapResultCollection(aux);
    }
    // ...}
```

Fuzzy query criteria instead of crisp ones can be introduced by extending the `Criteria` class and associated code. This can be accomplished alternatively by introducing additional methods to `Criteria` or by subclassing it. Fuzzy query results can be processed by casting to a class representing the pairs, resulting in a programming idiom like that described in [12]:

```
it = e.fuzzyIterator();
while (it.hasNext()){
   FuzzyObject aux = (FuzzyObject) it.next();
   X anX = (X) aux.getObject();
   double mu = aux.getMembership();
   // do processing...
 }
```

4 Example: Adding Fuzzy Classes and Fuzzy Numbers

The general design issues provided in the previous section can be used to implement a wide range of fuzzy extensions. In this section, we focus on two simple extensions for the sake of illustration, providing some important implementation details. Concretely, we will extend `OJB` with fuzzy classes and fuzzy numbers as attributes of classes, so that simple flexible queries can be issued through standard means. The domain used as a case study is that of market segmentation under fuzziness, taken from [11]. A conceptual model for the basic definitions in the case study is provided in Figure 2.

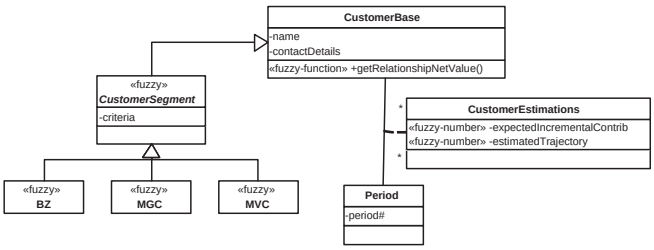


Figure 2: Main conceptual elements

In Figure 2 customers are instances of `CustomerBase`, and the estimated value of their relationship with the company is described from the marketing perspective in terms of estimations about duration (loyalty) and estimated increase in purchase volume for each period in the medium–term forecast. Both estimations are imprecise *fuzzy numbers* as marked by the stereotype `<<fuzzy-number>>`. Triangular fuzzy numbers can be represented by triples of real numbers (a, b, c) where $a < b < c$ and it is assumed that it represents a vague notion of real number roughly as "between a and c, and very close to b". The net value of their relationship is computed from those estimations by an algorithm producing also imprecise results (`<<fuzzy-function>>`), so it returns fuzzy numbers as return values. Such values is used by a process of fuzzy clustering not covered here that produce *fuzzy classes* BZ (below zero), MGC (most growable customers) and MVC (most valuable customers). Such classes are defined by overlapping membership functions as described in [11].

The system then uses the values for a sequences of periods to compute the membership degree of each customer for each of the classes (as in [11]), storing it in a explicit, extensional way, since marketing experts are able to change them due to other factors that may affect the relationship trajectory.

The storage of the example is used by specifying an XML schema like the following:

```
<class-descriptor  class="CustomerEstimations"
     table="CUSTOMER_EST" >
   <field-descriptor
   name="expectedIncContrib"
   column="EXP" left="EXP0" right="EXP1"
   type="FuzzyNum" att-left="left"
   att-center="center" att-right="right"
    primarykey="false" />
   ...
</class-descriptor>
<class-descriptor class   class="MGC"
  table="MGC" type="fuzzy" extensional="true"
  membershipTable="CUSTOMER_BASE" membField="MGC">
   ...
</class-descriptor>
```

Basically, schema definitions are `OJB`–like schemas with extended data mapping attributes and elements intended to be processed by `loadFuzzyDescRepository`. Fuzzy attributes (numbers) are mapped in the simplest way, by explicitly declaring the properties of the class that hold the left, center and right points describing the fuzzy number, so that the appropriate `getX()` methods could be invoked through reflection in the access layer.

Queries can be issued to the persistence layer by standard means provided in the core interfaces. For example, the following query returns a fuzzy subset of "BZ" (a subset of the crisp class `CustomerBase`) that has values (approximately) greater than 4.3.

```
broker.wrapFuzzySubsets();
Criteria criteria = new Criteria();
criteria.addGreaterOrEqualThan(
  "expectedIncContrib", new Double(4.3));
QueryByCriteria query = new QueryByCriteria(
        CustomerBase.class, criteria, "BZ");
Collection res=broker.getCollectionByQuery(query);
```

It should be noted that higher–level interfaces like `Jdo` can be used alternatively to carry out fuzzy queries. For example, the following `Jdo` query returns the fuzzy subset of MVC filtered (in a crisp way) by state and area.

```
String filter =
   "contactDetails.state == state && " +
   "contactDetails.area > area";
Extent extent = pm.getExtent(CustomerBase.class,
 true, "asc;fuzzySubset=MVC");
Query query = pm.newFuzzyQuery(extent, filter);
((FuzzyQuery)query).interpretAllFuzzy();
```

```
... Collection result =
(Collection)query.execute( "Georgia", "200");
```

In the above example, the "MVC" string encoded in the parameters to `getExtent` is used to specify the class descriptor associated to the actual Java class `CustomerBase`, and the `interpretAllFuzzy()` method explicitly forces the wrapping of query results for membership processing.

5 Conclusions

Fuzziness can be considered as a separate cross–cutting concern in existing software, and in consequence, AOD techniques provide a convenient framework to implement fuzzy extensions to existing libraries. As a proof of concept for such approach, the AOD extension of the `OJB` libraries using `aspectj` has been described, along with a concrete case study regarding implementations of class and attribute concerns for fuzziness. The resulting design combines inheritance and aspects to come up with an extension that entails no modifications to existing `OJB` source code. Concretely, metadata processing is weaved at initialization and querying times, and polymorphism is used to provide alternate metadata and query result processing idioms that handle fuzziness, achieving full backwards compatibility with existing code.

References

[1] Atkinson, M. P., Daynes, L., Jordan, M.J., Printezis, T., Spence, S.: An Orthogonally Persistent Java. *ACM Sigmod Record*, 25(4), 1996

[2] Bosc, P., Pivert, O.: Fuzzy Querying in Conventional Databases, In: Zadeh, L., Kacprzyk, J. (eds.): *Fuzzy Logic for the Management of Uncertainty*. John Wiley, New York, 1992, 645–671

[3] Buckles, B.P., Petry, F.E: A Fuzzy Representation of Data for Relational Databases. *Fuzzy Sets and Systems* 7, 1982, 213–226

[4] Chen, G.: *Fuzzy logic in data modeling : semantics, constraints, and database design*. Kluwer Academic Publishers, 1998

[5] De Caluwe, R. (ed.): *Fuzzy and Uncertain Object-Oriented Databases, Concepts and Models*. World Scientific, Singapore, 1997

[6] Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J. and Griswold, W.G.: An Overview of AspectJ. In: *Proc. of the European Conference on Object-Oriented Programming (ECOOP)*, 2001

[7] Russell, C. et al.: *Java Data Objects (JDO) Version 1.0*, proposed final draft, Java Specification Request JSR000012, 2001

[8] Sicilia, M. A., García, E., Gutiérrez, J. A.: Integrating fuzziness in object oriented modelling languages: towards a fuzzy-UML. In: *Proceedings of the International Conference on Fuzzy Sets Theory and its Applications (FSTA)*, 2002, 66-67

[9] Sicilia, M.A., García, E., Díaz, P. and Aedo, I.: Extending Relational Data Access Programming Libraries for Fuzziness: The fJDBC Framework. *Lecture Notes in Computer Science* 2522, Springer, 2002, 314–328

[10] Sicilia, M.A., Gutiérrez, J.A., García, E.: Designing Fuzzy Relations in Orthogonal Persistence Object-Oriented Database Engines. *Lecture Notes in Computer Science* 2527, Springer, 2002, 243-253

[11] Sicilia, M.A., García, E.: On Fuzziness in Relationship Value Segmentation: Applications to Personalized e-Commerce. *ACM SIGECOM Newsletter*, 4(2), 2003, 1–10

[12] Sicilia, M.A., García, E., Gutiérrez, J.A.: Introducing Fuzziness in Existing Orthogonal Persistence Interfaces and Systems. In: *Advances in Fuzzy Object-Oriented Databases: Modeling and Applications*, IDEA Group Publishing, 2004, 241–268

[13] Smets, P.: Imperfect information: Imprecision-Uncertainty. In: *Uncertainty Management in Information Systems: From Needs to Solutions*. Kluwer Academic Publishers, 1997, 225–254

[14] Sutton Jr., S.M. and Rouvellou, I.: Modeling Software Concerns in Cosmos. In *Proceedings of the First International Conference on Aspect–Oriented Software Development* (AOSD 2002), ACM Press, 127-133

[15] Sutton Jr, S.M. and Rouvellou, I.: Applicability of Categorization Theory to Multidimensional Separation of Concerns. In: *Proceedings of the Workshop on Advanced Separation of Concerns*, OOPSLA 2001

[16] Sutton Jr., S. M. and Tarr, P.: Aspect-Oriented Design Needs Concern Modeling. In: *Proc. of the Aspect Oriented Design Workshop on Identifying, Separating and Verifying Concerns in the Design*, Enschede, The Netherlands, 2002

[17] Yazici, A., George, R., Aksoy, D. (1998). Design and Implementation Issues in the Fuzzy Object-Oriented Data Model. *Information Sciences*, 108(1-4), 241–260

[18] Zadrozny, S., Kacprzyk, J: FQUERY for Access: Towards Human Consistent Querying User Interfaces. In: *Proceedings of the 1996 ACM Symposium on Applied Computing (SAC'96)*, 1996, 532–536

Scientific Formats for Object-Relational Database Systems: A Study of Suitability and Performance

Shirley Cohen[1], Patrick Hurley[2], Karl W. Schulz[2], William L. Barth[2], and Brad Benton[3]

[1]Computer and Information Science
University of Pennsylvania
shirleyc@cis.upenn.edu

[2]Texas Advanced Computing Center
The University of Texas at Austin
{phurley,karl,bbarth}@tacc.utexas.edu

[3] IBM Corporation
brad.benton@us.ibm.com

ABSTRACT

Commercial database management systems (DBMSs) have historically seen very limited use within the scientific computing community. One reason for this absence is that previous database systems lacked support for the extensible data structures and performance features required within a high-performance computing context. However, database vendors have recently enhanced the functionality of their systems by adding object extensions to the relational engine. In principle, these extensions allow for the representation of a rich collection of scientific datatypes and common statistical operations. Utilizing these new extensions, this paper presents a study of the suitability of incorporating two popular scientific formats, NetCDF and HDF, into an object-relational system. To assess the performance of the database approach, a series of solution variables from a regional weather forecast model are used to build representative small, medium and large databases. Common statistical operations and array element queries are then performed using the object-relational database, and the execution timings are compared against native NetCDF and HDF operations.

1. Introduction

A common approach taken in early scientific computing algorithms was to utilize flat, sequential files for the handling of input and output data. While these sequential files were simple to access, they had performance drawbacks for storing and accessing large datasets [1]. In addition, the use of binary representations caused interoperability issues because of the differing byte-order assumed on "big-endian" and "little-endian" platforms. Consequently, the scientific demands for an efficient, binary independent, and extensible method for reading and writing scientific data led to the development of specific data-access libraries. Two popular libraries designed to address these needs are the Network Common Data Form (NetCDF) [2] and the Hierarchical Data Format (HDF) [3]. Both formats include data structures for supporting large datasets, multi-dimensional arrays, a variety of data types (integers, floating point numbers, strings), and methods to accommodate metadata. Both libraries also provide data access methods through an application programming interface that is available in several languages including Fortran, C, C++, and Java, which allow researchers to integrate NetCDF or HDF into their programs with relative ease.

While tools such as NetCDF and HDF offer a number of benefits over flat sequential files, the scientific community still frequently contends with the movement and storage of a large number of individual solution files. Indeed, researchers often output at least one solution file per simulation and then compute metrics by looping over all the files for a particular problem definition (e.g. multiple time-steps in a time-accurate physical simulation). One advantage of utilizing a database management system (DBMS) is that the entire solution set can be housed within one centralized location. Although this approach may seem to be an obvious solution, the relational model's lack of previous support for multi-dimensional arrays and poor performance on large, array-based datasets has limited its scientific applicability [4]. However, the emergence of object-relational database systems [5] has opened the door for including scientific datasets within a DBMS. Consequently, this paper explores an initial suitability study by considering the following questions:

- Is it possible to use an object-relational database as a storage structure for scientific datasets?
- If so, how does the data access performance using a relational engine compare to the performance of native NetCDF and HDF operations?

To examine these questions, several Oracle10g object-relational databases are derived using sample scientific datasets obtained from the Weather Research and Forecasting (WRF) Model. To construct the databases, existing SQL extensions are utilized to develop a physical schema that supports the storage and access of multi-

dimensional arrays. As such, the ideas introduced in this paper are generally applicable in the sense that they can be adapted to other object-relational DBMSs.

2. Scientific Data and the WRF Model

The WRF application is a community-based mesoscale weather prediction model originally developed at NCAR [6]. As a scientific application, the WRF model makes extensive use of the NetCDF format to define both input boundary conditions and output forecast solutions. In order to characterize a typical scientific workflow using WRF, a series of regional, 12-hour forecasts at a 40km resolution were generated spanning one month from July, 1995. The result of these simulations was the creation of over 400 NetCDF solution files which aggregate to provide approximately 5GB of time-dependent, gridded meteorological data. This sample WRF dataset was chosen for evaluation based on its heavy utilization of multi-dimensional array structures and native use of the NetCDF file format. In addition, it includes a modest number of solution files and within the context of climate modeling, these files are often post-processed in a serial fashion to derive specific statistical metrics (e.g., the average surface temperature at 1pm for the month).

Each individual solution file contains 77 floating point, multi-dimensional variables. However, a subset of three specific variables was identified for consideration in this study: accumulated precipitation, soil moisture, and temperature. These three variables are representative of the different array sizes and dimensionality present in the WRF output. Note that the precipitation variable is the smallest of the three and represents a two-dimensional spatial quantity which spans the lateral grid dimensions. In contrast, the temperature variable is the largest array and spans all three spatial dimensions (two lateral and one vertical). The soil moisture variable is sized in the middle and is dimensioned as a two-dimensional surface quantity for each of the modeled soil layers. In addition to the spatial dimensionality described for these variables, each variable is also a function of time. Consequently, the NetCDF representation for these variables defines the precipitation as a three-dimensional array, and the soil moisture and temperature variables as four-dimensional arrays.

3. Modeling Scientific Data

The three floating-point variables considered from the WRF dataset are defined as an array of values in which every element is associated with a unique time and space dimension. From the database perspective, the standard way of representing structures in a relational database is to use a star or snowflake schema [7] where each dimension is decomposed into its own table and is linked to the dependent variable using a foreign key relation. In an object-relational database, however, the dependent variable can be represented in a more natural manner. Rather than using a foreign key relation, the dependent variable can be stored with its dimensions using a multi-dimensional array column implemented as an abstract data type. The DBMS then keeps track of the mappings between the abstract data type and the other array dimensions that are implemented as regular table columns.

Commercial databases such as Oracle 10g, DB2 UDB 8, and SQL Server 2005 have adopted an object-relational data model that supports relational tables, abstract data types, and functions on abstract data types [8]. To simplify coding aspects, Oracle also provides two built-in array types that are commonly referred to as collections: Variable Arrays (Varray) and Nested Tables [9]. These two array types can be exploited to model scientific data. For the purpose of this study, Oracle 10g was chosen to store and operate on the multi-dimensional variables derived from the WRF dataset. Note that both types are stored in relational tables and can be used as a table column or an attribute of an object type. A Varray has fixed lower and upper bounds, similar to collection types from other programming languages. It stores elements in the order in which they are added, while a Nested Table does not preserve the insertion order. One additional difference is that Nested Tables may be indexed, while indexing a Varray is not permitted. Nested Tables have another advantage over Varrays in that they allow for inserts, updates, and deletes on individual elements. The Varray type does not allow such operations since Varray data is stored as a single, delimited piece of data within the database. To illustrate the hierarchical relationships between the Table and Varray types, and Table and Nested Table types, Figure 1 presents a schema diagram for the temperature variable. The syntax for the create type and table statements are presented in Figure 2. Note that the NUMBER(9,3) defines a floating-point number with a precision of 9 and a scale of 3. Except for the names used, the schema diagrams and the CREATE statements for soil moisture and precipitation are identical.

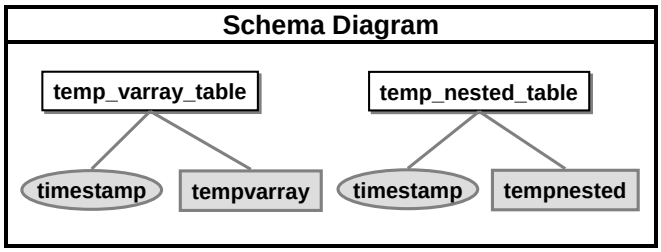


Figure 1: Schema diagram illustrating the hierarchical structure of the relationship types.

Varray

```
CREATE OR REPLACE TYPE tempvarray AS
VARRAY (1000000) OF NUMBER(9,3);

CREATE TABLE temp_varray_table(
timestamp date, data tempvarray);
```

Nested Table

```
CREATE OR REPLACE TYPE tempnested AS TABLE
OF NUMBER(9,3);

CREATE TABLE temp_nested_table(
TIMESTAMP date, data tempnested)
NESTED TABLE data STORE AS data_tab_temp;

CREATE INDEX temp_index ON
data_tab_temp(column_value);
```

Figure 2: Oracle CREATE type and table statements for the temperature variable.

4. Preparing Scientific Data

Scientific computing tasks can easily generate many gigabytes, if not terabytes, of data per run. Once generated, the data must be extracted from multiple files, cleaned, and transformed before it can be loaded into a database. Additional preprocessing steps may also be required depending on the application: e.g. checking integrity constraints, sorting, summarization, and building indices. This preprocessing step can be a time-consuming effort although batch loading utilities can greatly facilitate the process. In this study, Oracle's SQL*Loader utility was used to populate the object-relational schema described in the previous section with the temperature, soil moisture, and precipitation variables. Although the WRF dataset contained additional variables, we chose to load only the three variables of interest for convenience. To use the SQL*Loader utility, two input files were required: a control file that specified how data should be loaded into the database, and a data file, which specified the data to be loaded.

To accommodate performance measurements for different sized datasets, three unique databases were constructed using the 403 NetCDF files included in the WRF dataset. The databases are categorized into small, medium, and large sizes and their designation is based on the number of NetCDF files ingested. The large database includes variables from all 403 files, while the medium and small databases include 195 and 91 files, respectively. To illustrate the number of floating-point elements included in each of these datasets, Table 1 presents the total number of elements included within each of the three variable arrays for each dataset classification. The platform used to construct these Oracle databases was a uni-processor Linux server running RedHat 9 Enterprise with a 3.06GHz Xeon processor and 2GB of main memory. All data was stored on a local, ext3 file system. The 10g Enterprise Edition of Oracle was used (*Release 10.1.0.3.0*), and it was configured with the default 16MB buffer pool.

# of Array Elements			
Dataset	**Temp**	**Soil**	**Pressure**
Large	79,804,075	12,768,652	3,192,163
Medium	38,614,875	6,178,380	1,544595
Small	18,020,275	2,883,244	720,811

Table 1: Number of floating-point elements per variable stored in the small, medium, and large datasets.

To quantify the amount of time required to build the scientific databases, Table 2 presents the wall-clock time required to load each of the three meteorological variables for all three database sizes. Note that during the course of the loading process, several flaws were encountered with the SQL*Loader utility. Initially, none of the data was properly ingested due to an underscore character in the name of the Varray and Nested Table type object definition. Additionally, all records that were larger than 1MB were rejected, which in our case, constituted the majority of the records. Both problems prevented successful creation of the databases, and required several exchanges with Oracle Support. Based on these discussions, subsequent software patches were provided which eventually circumvented the errors. Note that the load times for the nested table were not inconsequential for the larger variables. In particular, it required over 30 minutes to load the temperature variable into the large database. In contrast, the same operation using Varrays required less than 7 minutes to complete.

Varray Load Times (mm:ss)			
Variable	**Database Size**		
	Large	**Medium**	**Small**
Temperature	06:17	02:51	01:21
Soil Moisture	00:43	00:18	00:08
Precipitation	00:12	00:05	00:02
Nested Table Load Times (mm:ss)			
Variable	**Database Size**		
	Large	**Medium**	**Small**
Temperature	31:41	15:11	07:19
Soil Moisture	04:56	02:18	01:02
Precipitation	01:08	00:29	00:06

Table 2: Wall-clock time required to load three WRF dataset variables into large, medium, and small databases.

5. Querying Scientific Data

Once the Varrays and Nested Tables were created and populated, queries were formulated using standard SQL to compute common statistical quantities of interest. To illustrate these queries, Figure 3 presents the syntax required to compute the sum, average, minimum, and maximum values for the temperature variable on Varrays using standard SQL-92 aggregation functions.

In addition to aggregate queries, application scientists are often interested in accessing a small subset of a particular array. Oracle provides no built-in methods for retrieving an element by its array index. To provide this capability, we defined two user-defined functions as presented in Figure 4. As input, the function *getIndexValueVarray* takes a Varray structure and an index and returns the value of the array element corresponding to the provided index. The function *getIndexValueNested* performs the same operation on a Nested Table. Once created, these functions can be invoked using standard SQL. Example syntax demonstrating the use of these functions is shown in Figure 5. These example queries apply a WHERE clause to identify records with a timestamp of "1995-07-01 00". The records that satisfy this filter are passed to the *getIndexValueVarray* and *getIndexValueNested* operators. The results from these operators are instances of the Varray and Nested Table abstract datatypes, respectively. Though their structures are fixed, these operators were written to handle any array variable stored in a Varray or Nested Table type. In addition to these examples, the object-relational model allows for the implementation of more complex analytical functions inside the DBMS.

6. Performance Study

The four aggregate queries and three index queries discussed in the previous section were run to evaluate the performance of Varrays and Nested Tables. The approach chosen to evaluate the database queries was based on an analysis of the execution plans generated by Oracle's cost based optimizer [10, 11]. Additionally, the I/O performance of each test was evaluated using *iostat*, a standard Linux utility for monitoring system I/O performance. For performance comparisons, NetCDF 3.6.0-p1, HDF 4.2r1, and HDF 5 were used. The data was originally in NetCDF format and converted into HDF format using an open-source utility [12].

All of the queries were run when the test machine was idle and the results are based on the average execution time of five successive runs. Note that one of the main goals during these tests was to minimize the caching effects on each measurement; this was accomplished by adhering to a strict set of testing sequences as presented in Figure 6. The purpose of unmounting the file system between each query was to flush out the operating system's buffer cache. The remount of the file system was followed by running a simple program which allocated and initialized a large portion of the host's memory in order to clear out the second-level cache. Using this approach, we were able to obtain consistent timings which minimize any file or memory related caching. Note, however, that in a production environment, the measured response times of repeated queries could be much lower than the average values presented herein due to caching benefits.

Varray Query

```
SELECT SUM(t2.COLUMN_VALUE)
FROM temp_varray t1, TABLE(t1.data) t2;

SELECT AVG(t2.COLUMN_VALUE)
FROM temp_varray t1, TABLE(t1.data) t2;

SELECT MIN(t2.COLUMN_VALUE)
FROM temp_varray t1, TABLE(t1.data) t2;

SELECT MAX(t2.COLUMN_VALUE)
FROM temp_varray t1, TABLE(t1.data) t2;
```

Figure 3: SQL query syntax to derive statistical metrics of the temperature variable using Varrays.

```
CREATE or REPLACE function
getIndexValueVarray (data VARRAY,
array_index number)
RETURN number is
BEGIN
RETURN data(array_index);
END;

CREATE or REPLACE function
getIndexValueNested (nestedtab typenested,
array_index number)
RETURN number is
BEGIN
RETURN nestedtab(array_index);
END;
```

Figure 4: User-defined functions to access individual array elements.

```
SELECT getIndexValueVarray(sn.data,
198025) FROM temp_varray sn
WHERE timestamp = to_date('1995-07-01 00',
'yyyy-mm-dd HH24');

SELECT getIndexValueNested(sn.data,
198025) FROM TEMP_NESTED sn
WHERE timestamp = to_date('1995-07-01 00',
'yyyy-mm-dd HH24');
```

Figure 5: Example SQL syntax to query individual array elements using Varrays and Nested Tables.

Oracle Database	Scientific File Formats
1. Shutdown Oracle 2. Unmount file system 3. Mount file system 4. Fill up memory 5. Startup Oracle 6. Flush buffer cache 7. Run query	1.Unmount file system 2.Mount file system 3.Fill up memory 4.Run Query

Figure 6: Testing sequences for the Oracle and scientific file format queries.

6.1 Aggregate Queries

Table 3 presents the measured query times for the small, medium, and large datasets. Note that four aggregate queries (sum, average, minimum, and maximum) and three index queries (first, middle, and last element) were run on each of the datasets. However, the results obtained for each dataset were almost identical for both the aggregate queries and the index queries. Due to this small variation, we thus include only one aggregate query (average) and one index query (middle) in the results.

Aggregate Query Timings (*in secs*)					
Large Dataset					
Variable	**NetCDF**	**HDF4**	**HDF5**	**Varray**	**Nested**
Temp.	16.56	67.11	20.99	1472.24	40.08
Soil	13.39	61.59	16.28	87.07	6.33
Press.	10.51	55.14	14.87	17.26	1.38
Medium Dataset					
Variable	**NetCDF**	**HDF4**	**HDF5**	**Varray**	**Nested**
Temp.	7.60	31.22	10.27	700.54	20.89
Soil	6.41	28.82	7.97	42.40	3.92
Press.	4.92	25.53	7.04	8.52	1.29
Small Dataset					
Variable	**NetCDF**	**HDF4**	**HDF5**	**Varray**	**Nested**
Temp.	3.48	14.20	4.64	327.48	10.21
Soil	2.97	13.01	3.63	19.11	2.17
Press.	2.18	11.58	3.16	4.11	0.62

Table 3: *Aggregate Query* - wall-clock measurements of the time required to compute an average value for the temperature, soil, and precipitation variables from small, medium, and large datasets.

For all three datasets, the HDF, NetCDF, and Nested Table methods outperformed Varrays on the large temperature and medium soil variables by as much as 90%. The main factor contributing to this performance difference is the use of an expensive nested-loop join [13] in which Oracle fetches a batch of timestamps from the outer Varray table before probing the inner Varray type. In contrast, the Nested type is directly accessible by the query processor, thus avoiding the cost of the join operation. For this reason, performing a full table scan operation on the Varrays is more expensive than on the Nested Tables.

Individual Index Query Timings (*in secs*)					
Large Dataset					
Variable	**NetCDF**	**HDF4**	**HDF5**	**Varray**	**Nested**
Temp.	0.016	0.010	0.020	0.170	40.180
Soil	0.014	0.010	0.005	0.090	6.370
Pressure	0.015	0.009	0.003	0.070	1.340
Medium Dataset					
Variable	**NetCDF**	**HDF4**	**HDF5**	**Varray**	**Nested**
Temp.	0.016	0.010	0.020	0.160	20.200
Soil	0.017	0.009	0.005	0.070	3.750
Pressure	0.015	0.009	0.003	0.060	1.200
Small Dataset					
Variable	**NetCDF**	**HDF4**	**HDF5**	**Varray**	**Nested**
Temp.	0.016	0.010	0.020	0.160	10.220
Soil	0.014	0.010	0.005	0.080	1.970
Pressure	0.015	0.009	0.003	0.050	0.070

Table 4: *Index Query* - wall-clock measurements of the time required to read the middle array value for the temperature, soil, and precipitation variables from small, medium, and large datasets.

The results also suggest that the database scales less than the file-based approach with the larger variables. For example, Nested Tables were more than 3 times faster on the small precipitation variable than the other methods, while the same queries on the large temperature variable took about twice as long as the netCDF and HDF queries. For the temperature variable queries, we note that Oracle was reading up to five times as much data from the disk as the file-based methods. The additional reads incurred by Oracle could be due to a less efficient data representation and a higher ratio of metadata per query.

On the precipitation variable, the database was transferring about the same amount of data as netCDF and over twice as much data as HDF. In this case, Oracle was able to coalesce its I/Os into larger and fewer transactions overall, which explains the faster execution times on the Nested Table queries with the precipitation variable.

6.2 Index Queries

For the index queries, a similar trend emerges in which both Varrays and Nested Tables performed worse than the file-based approaches as the number of elements increased per variable. In this case, however, the Nested queries performed significantly worse than the Varrays, by as much as 200%. The execution plans indicate that Nested Table queries require a full table scan to retrieve a single index. This explains why the Nested index numbers are similar to the aggregates. In total, Varray

index queries resulted in 5 fewer database blocks fetched from disk than the Nested queries. For the small precipitation variable, the Varrays had fewer I/O requests than both HDF methods and NetCDF. With the temperature variable, however, NetCDF required fewer I/O requests than the other methods. Consequently, the execution time correlates with the amount of data read.

7. Conclusion

This paper examined an object-relational DBMS through the prism of scientific data management. The extensibility of object-relational features such as abstract datatypes and user-defined functions were matched to the datatypes and operations commonly employed within the general scientific community. In particular, abstract data types were used to compare the performance of multi-dimensional array structures inside an object-relational DBMS versus native HDF and netCDF file access methods. These performance comparisons were done using Oracle 10g because of its inclusion of two convenient built-in array types: Varray and Nested Tables. We conclude that the database performed well as long as the number of elements per array was small, but did not scale as well as the scientific file-based methods when the number of elements grew. Nested Tables outperformed all the other methods on the small precipitation variable for the aggregate queries, but was slower than the scientific file formats on the larger temperature variable. Similarly, with the index queries, Varrays performed well overall for the smaller precipitation variable, but were less efficient with the larger temperature variable. These results suggest that for the schema considered, the support for array-based object-relational technology from a standard Oracle10g installation is at best mixed and there is substantial room for improving usability and performance. In addition, Oracle10g's storage structure and query processing can most likely fit the needs of small multi-dimensional arrays with fewer than 1 million elements as long as the user is willing to accept the burdens of installing the DBMS, designing and creating the schema, and loading the data into the database. Our overall experience with Oracle 10g indicates that none of these tasks are trivial (especially for a typical scientific user who likely lacks experience in DBMS administration).

The authors would like to note that the work reported here is a proof-of concept investigation which opens the door for further research. In particular, there may be additional tuning possibilities for Oracle10g to increase performance and other DBMSs may behave differently. Another issue to consider is how exactly should these scientific arrays interact with the SQL engine? Since arrays are implemented as abstract datatypes, they are not transparent to the query optimizer. Further work is likely needed to exploit statistics for object-relational DBMSs and produce higher quality query plans. Other issues to be resolved include developing techniques for parallelizing arrays and their operators so that they can scale to larger datasets.

8. Acknowledgements

We wish to thank Laura Timm and Chris Hempel for their help with the Oracle 10g installation. We also thank Tommy Minyard and John R. Boisseau for discussions on scientific data management.

9. References

[1] Gray, J., Liu, D., Nieto-Santisteban, M., Szalay, A., DeWitt, D., and Heber, G., *Scientific Data Management in the Coming Decade*. Microsoft Research Technical Report MSR-TR-2005-10 (2005).

[2] Rew, R. K., G. P. Davis, S. Emmerson, and H. Davies, "NetCDF User's Guide for C, An Interface for Data Access**",** Version 3, April 1997.

[3] Hierarchical Data Format (HDF): http://hdf.ncsa.uiuc.edu/hdf4.html

[4] Network Common Data Form – NetCDF is Not a Database Management System: http://my.unidata.ucar.edu/content/software/netcdf/docs/netcdf/Not-DBMS.html

[5] Stonebraker, M. and Brown, P., Object-Relational DBMSs: Tracking the Next Great Wave, 2nd edition. San Francisco: Morgan Kaufmann Publishers Inc., 1999.

[6] Michalakes, J., J. Dudhia, D. Gill, T. Henderson, J. Klemp, W. Skamarock, and W. Wang, "The Weather Research and Forecast Model: Software Architecture and Performance", Proceedings of the 11th ECMWF Workshop on the Use of High Performance Computing In Meteorology, October, 2004.

[7] Kim, Won. "Modern Database Systems: The Object Model, Interoperability and Beyond." ACM Press and Addison-Wesley, 1995.

[8] Krishnamurthy, V., Banerjee, Sandeepan, Nori, Anil. "Bringing Object-Relational Technology to Mainstream." SIGMOD Conference 1999: 513-514.

[9] Oracle Database. Application Developer's Guide – Object Relational Features 10g Release 1 (10.1). December 2003.

[10] Oracle Corporation. Cost Based Optimizer (CBO) Overview. Note: 10626.1. June 2002.

[11] Oracle Corporation. Cost Based Optimizer – Common Misconceptions and Issues. Note: 35934.1. April 2004.

[12] The NetCDF2HDF conversion utility: http://ioc.unesco.org/oceanteacher/resourcekit/M3/Converters/NetCDF2HDF/.

[13] Oracle Corporation. Interpreting Explain plan. Note 46234.1. October, 2003.

Temporal Aggregates and Temporal Universal Quantification in Standard SQL

Esteban Zimányi
Dept. of Computer & Network Engineering
Université Libre de Bruxelles, CP 165/15
50 Av. F. Roosevelt, 1050 Bruxelles, Belgique
ezimanyi@ulb.ac.be

ABSTRACT

Although it has been acknowledged for many years that querying and updating time-varying information using standard (i.e., non-temporal) SQL is a challenging task, the proposed temporal extensions of SQL have not reach acceptance in the standardization committees. Therefore, nowadays database practitioners must still use standard SQL for manipulating time-varying information. This paper shows how to realize temporal aggregates and temporal universal quantifiers using standard SQL.

Keywords: Temporal databases, temporal aggregates, temporal universal quantifiers, SQL.

1. INTRODUCTION

Many applications need to keep the evolution of data that varies on time. For example, in order to conform with legal regulations and corporate policies, administrative databases must keep the evolution of many attributes of their employees, such as marital status, salary, affiliation, etc. Current Database Management Systems, and SQL in particular, provide little support for dealing with time-varying data: They only provide standard data types for encoding dates or timestamps. However, querying and updating time-varying data using standard SQL is a challenging task.

For this reason, in the last decades many research was devoted to the management of the temporal dimension in databases, resulting in the development of temporal databases. In particular, a temporal extension of SQL-92 called TSQL2 [7] was proposed to international standardization committees [9, 10] leading to a dedicated chapter of the SQL:1999 standard called "Part 7, SQL/Temporal". However, such an extension has not yet passed the standardization process [11, 2]. Another approach for coping with the temporal dimension in relational databases was proposed in [3].

The consequence of this state of affairs is that nowadays database practitioners are left with standard SQL for manipulating time-varying data. In [8] is presented how to manipulate temporal data with standard SQL, discussing in which situations and under which assumptions the corresponding SQL code can be applied. In particular, it is presented how to define temporal join, temporal projection, or temporal difference in SQL. However, at the best of our knowledge, there has not been studies showing how to deal with temporal aggregates as well as temporal universal quantification using standard SQL. This paper is devoted to this issue.

The paper is structured as follows. Section 2, inspired from [8], introduces temporal databases and shows how to realize temporal join and temporal projection in standard SQL. The section provides the necessary background for the rest of the paper. Sections 3 and 4, describing our contributions, are devoted to temporal aggregates and temporal universal quantification. Finally, in the conclusion we describe experimental results and comparison with related works.

2. TEMPORAL DATABASES

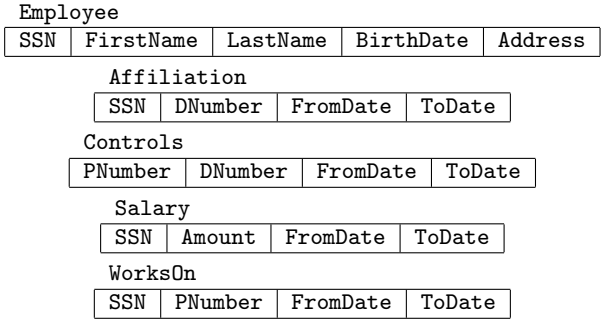


Figure 1: Example of a temporal database schema.

Figure 1 shows an example of a temporal database schema. Table `Employee` is non-temporal, while all other tables are temporal, and more precisely, they are valid-time tables: The columns `FromDate` and `ToDate` indicate when the information in the corresponding row is valid, e.g., the period of time during which an employee is affiliated to a department. As a data type for representing periods is not available in SQL, a period is encoded in two columns of type `Date`. The problem is that these peculiar columns (i.e., having such a particular semantics) are encoded in SQL in the same way as columns such as `BirthDate` which is also of type `Date` but does not have any specific semantics.

SSN	DNumber	FromDate	ToDate
123456789	D1	2002-01-01	2003-06-01
123456789	D1	2003-06-01	2003-08-01
123456789	D1	2003-08-01	3000-01-01
333444555	D2	2003-10-01	2004-01-01
333444555	D2	2004-01-01	3000-01-01

Figure 2: An example of temporal table.

Figure 2 shows an example of table `Affiliation`. A closed-open representation for periods is used, e.g., the va-

lidity of the first row is [2002-01-01, 2003-06-01). Also, a special date '3000-01-01' denotes currently-valid rows.

2.1 Temporal Join

The join operator is used to combine information scattered in different tables. If the tables to be combined are temporal, then a temporal join is needed. Expressing a temporal join in SQL requires four select statements and complex inequality predicates verifying that the validity periods of the rows to be combined intersect.

Recall from Figure 1 that tables Salary and Affiliation keep, respectively, the evolution of the salary and the evolution of the affiliation of employees. To determine the joint evolution of salary and affiliation a temporal join of these tables is needed. This can be done in SQL as follows.

```
SELECT S.SSN,Amount,DNumber,S.FromDate,S.ToDate
FROM Salary S, Affiliation A WHERE S.SSN=A.SSN
AND A.FromDate<S.FromDate AND S.ToDate<=A.ToDate
  UNION ALL
SELECT S.SSN,Amount,DNumber,S.FromDate,A.ToDate
FROM Salary S, Affiliation A
WHERE S.SSN=A.SSN AND S.FromDate>=A.FromDate
AND S.FromDate<A.ToDate AND A.ToDate<S.ToDate
  UNION ALL
SELECT S.SSN,Amount,DNumber,A.FromDate,S.ToDate
FROM Salary S, Affiliation A
WHERE S.SSN=A.SSN AND A.FromDate>=S.FromDate
AND A.FromDate<S.ToDate AND S.ToDate<A.ToDate
  UNION ALL
SELECT S.SSN,Amount,DNumber,A.FromDate,A.ToDate
FROM Salary S, Affiliation A WHERE S.SSN=A.SSN
AND A.FromDate>S.FromDate AND A.ToDate<S.ToDate
```

In the above query it is supposed that there are no duplicate rows in the tables: at each point in time an employee has one salary and one affiliation. The UNION ALL is used since the query does not generate duplicates and this is more efficient than using UNION.

Temporal join can be written in a single statement using either a CASE statement or using functions. Suppose that two functions minDate and maxDate are defined as follows.

```
CREATE FUNCTION minDate(one DATE,two DATE)
RETURNS DATE BEGIN
  RETURN CASE WHEN one<two THEN one ELSE two END
END
CREATE FUNCTION maxDate(one DATE,two DATE)
RETURNS DATE BEGIN
  RETURN CASE WHEN one>two THEN one ELSE two END
END
```

Thus, minDate and maxDate return, respectively, the minimum and the maximum of the two arguments. Using the above functions the temporal join can be defined as follows.

```
SELECT S.SSN,Amount,DNumber,
  maxDate(S.FromDate,A.FromDate) AS FromDate,
  minDate(S.ToDate,A.ToDate) AS ToDate
FROM Salary S, Affiliation A WHERE S.SSN=A.SSN
AND maxDate(S.FromDate,A.FromDate) <
    minDate(S.ToDate,A.ToDate)
```

The two functions are used in the SELECT clause for constructing the intersection of the corresponding validity periods. The condition in the WHERE clause ensures that the two validity periods overlap.

2.2 Temporal Projection

SSN	FromDate	ToDate
123456789	2002-01-01	2003-06-01
123456789	2003-06-01	2003-08-01
123456789	2003-08-01	3000-01-01
333444555	2003-10-01	2004-01-01
333444555	2004-01-01	3000-01-01

Figure 3: Projecting DNumber from Figure 2.

Given the table Affiliation of Figure 2, suppose that we want to obtain the periods of time in which an employee has worked in the company, independently of the department. Figure 3 shows the result of projecting out attribute DNumber from the table of Figure 2. As can be seen the resulting table is redundant. The first three rows are value equivalent (i.e., they equal on all their columns but FromDate and ToDate) and the validity periods of these rows meet. The situation is similar for the last two rows. Therefore, the result of the projection should be as given in Figure 4. This process of combining several value-equivalent rows into one provided that their validity periods overlap is called coalescing.

SSN	FromDate	ToDate
123456789	2002-01-01	3000-01-01
333444555	2003-10-01	3000-01-01

Figure 4: Coalescing the temporal table of Figure 3.

Coalescing is a complex and costly operation in SQL. It can be realized entirely in SQL as follows [1].

```
SELECT DISTINCT F.SSN,F.DNumber,F.FromDate,L.ToDate
FROM Affiliation F, Affiliation L
WHERE F.FromDate<L.ToDate
AND F.SSN=L.SSN AND F.DNumber=L.DNumber
AND NOT EXISTS ( SELECT * FROM Affiliation M
  WHERE M.SSN=F.SSN AND M.DNumber=F.DNumber
  AND F.FromDate<M.FromDate AND M.FromDate<=L.ToDate
  AND NOT EXISTS ( SELECT * FROM Affiliation M1
    WHERE M1.SSN=F.SSN AND M1.DNumber=F.DNumber
    AND M1.FromDate<M.FromDate
    AND M.FromDate<=M1.ToDate ) )
AND NOT EXISTS ( SELECT * FROM Affiliation M2
  WHERE M2.SSN=F.SSN AND M2.DNumber=F.DNumber
  AND ( (M2.FromDate<F.FromDate AND
         F.FromDate<=M2.ToDate)
     OR (M2.FromDate<=L.ToDate AND
         L.ToDate<M2.ToDate) ) )
```

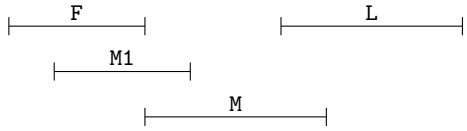


Figure 5: Coalescing value-equivalent rows.

Consider the diagram in Figure 5. Coalescing amounts to select the from and to dates of two value-equivalent rows F(irst) and L(ast) having no gap between these dates, i.e., for every value-equivalent row M whose validity period is between those of F and L there is another value-equivalent row

M1 whose validity period overlaps the beginning of M. The second NOT EXIST ensures that no other value-equivalent row M2 overlaps the period between the selected from and to dates and has an earlier from date or a later to date.

3. TEMPORAL AGGREGATION

SQL provides aggregation functions such as COUNT, MIN, MAX, and AVG. They are used for answering queries such as "List the maximum salary" or "Count the number of employees". If table Salary were non-temporal these queries can be written as follows.

```
SELECT MAX(Amount)          SELECT COUNT(*)
FROM Salary                 FROM Salary
```

Another usual request is to combine rows according to a criterion specified in a GROUP BY clause previous to the application of the aggregation operator, as in the query "List the maximum salary by department". The non-temporal version of this query can be written in SQL as follows.

```
SELECT DNumber,MAX(Amount)
FROM Affiliation A, Salary S
WHERE A.SSN=S.SSN GROUP BY DNumber
```

The temporal version of the above queries require a three-step process: (i) identify the periods of time in which all values are constant, (ii) compute the aggregation over these periods, and finally (iii) coalesce the result.

For the first two queries, Figure 6 shows a diagram where table Salary has three employees E1, E2, and E3, as well as the temporal maximum and temporal count. Notice that the period in which no employee works in the company does not belong to the answer for the temporal maximum, but it appears for the temporal count with a value of 0.

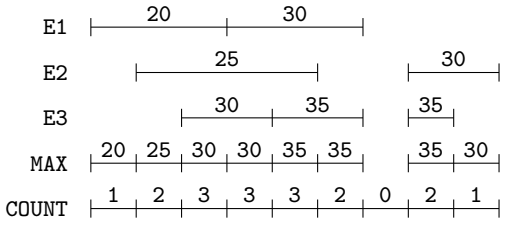


Figure 6: Evolution of the maximum salary and the number of employees.

The first step computes the periods on which the aggregation must be calculated, as follows.

```
CREATE VIEW SalChanges(Day) as
  SELECT DISTINCT FromDate FROM Salary
  UNION SELECT DISTINCT ToDate FROM Salary
CREATE VIEW SalPeriods(FromDate,ToDate) as
  SELECT P1.Day,P2.Day
  FROM SalChanges P1, SalChanges P2
  WHERE P1.Day<P2.Day AND NOT EXISTS (
    SELECT * FROM SalChanges P3
    WHERE P1.Day<P3.Day AND P3.Day<P2.Day )
```

View SalChanges gathers the days in which a salary change occurred, while view SalPeriods constructs the periods from such days.

The second step computes the aggregation on these periods. For the maximum salary this is done as shown below.

```
CREATE VIEW TempMax(MaxSalary,FromDate,ToDate) as
  SELECT MAX(Amount),P.FromDate,P.ToDate
  FROM Salary S, SalPeriods P
  WHERE S.FromDate<=P.FromDate
  AND P.ToDate<=S.ToDate
  GROUP BY P.FromDate, P.ToDate
```

Computing the number of employees is done as follows.

```
CREATE VIEW TempCount(NbEmp,FromDate,ToDate) as
  SELECT COUNT(*),P.FromDate,P.ToDate
  FROM Salary S, SalPeriods P
  WHERE S.FromDate<=P.FromDate
  AND P.ToDate<=S.ToDate
  GROUP BY P.FromDate, P.ToDate
UNION ALL
  SELECT 0,P.FromDate,P.ToDate FROM SalPeriods P
  WHERE NOT EXISTS ( SELECT * FROM Salary S
    WHERE S.FromDate<=P.FromDate
    AND P.ToDate<=S.ToDate )
```

Notice the second select that assigns a count value of 0 to those periods that do not appear in the first select.

Finally, in the third step it is necessary to coalesce the above views. This can be done as seen in Section 2.2.

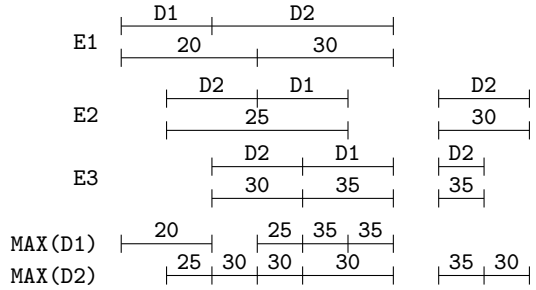


Figure 7: Maximum salary by department.

Now consider the second query asking the maximum salary by department. Figure 7 shows a diagram of possible values of tables Affiliation and Salary for employees E1, E2, and E3, and departments D1, and D2. If it is supposed that employees have salary only while they are affiliated to a department, the query can be written as follows.

In the first step it is necessary to compute by department the periods on which a maximum must be calculated.

```
CREATE VIEW Aff_Sal(DNumber,Amount,
    FromDate,ToDate) as
  SELECT DISTINCT A.DNumber,S.Amount,
    maxDate(S.FromDate,A.FromDate),
    minDate(S.ToDate,A.ToDate)
  FROM Affiliation A, Salary S WHERE A.SSN=S.SSN
  AND maxDate(S.FromDate,A.FromDate) <
      minDate(S.ToDate,A.ToDate)
CREATE VIEW SalChangesDep(DNumber,Day) as
  SELECT DISTINCT DNumber,FromDate FROM Aff_Sal
  UNION SELECT DISTINCT DNumber,ToDate FROM Aff_Sal
CREATE VIEW SalPeriodsDep(DNumber,FromDate,ToDate) as
  SELECT P1.DNumber,P1.Day,P2.Day
  FROM SalChangesDep P1, SalChangesDep P2
  WHERE P1.DNumber=P2.DNumber AND P1.Day<P2.Day
  AND NOT EXISTS ( SELECT * FROM SalChangesDep P3
    WHERE P1.DNumber=P3.DNumber AND P1.Day<P3.Day
    AND P3.Day<P2.Day )
```

View Aff_Sal realizes a temporal join of Affiliation and Salary. This yields the days in which a change of maximum salary of a department may occur. Next, view SalChanges collects such days by department, and finally view SalPeriods constructs the periods from these days.

The second step computes the maximum salary for the above periods.

```
CREATE VIEW TempMaxDep(DNumber,MaxSalary,
    FromDate,ToDate) as
  SELECT P.DNumber,MAX(Amount),P.FromDate,P.ToDate
  FROM Aff_Sal A, SalPeriodsDep P
  WHERE A.DNumber=P.DNumber
  AND A.FromDate<=P.FromDate AND P.ToDate<=A.ToDate
  GROUP BY P.DNumber, P.FromDate, P.ToDate
```

Finally, in the third step the above view is coalesced.

4. TEMPORAL UNIVERSAL QUANTIFIER

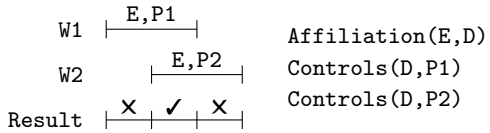
The universal quantifier is needed in many usual queries, such as "List the employees that work in **all** projects controlled by the department to which they are affiliated". As SQL does not provide the universal quantifier, the non-temporal version of the above query is written with two nested NOT EXISTS as follows.

```
SELECT SSN FROM Affiliation A WHERE NOT EXISTS (
  SELECT * FROM Controls C WHERE A.DNumber=C.DNumber
    AND NOT EXISTS ( SELECT * FROM WorksOn W
    WHERE C.PNumber=W.PNumber AND A.SSN=W.SSN ) )
```

Consider now the temporal version of the above query. As was the case for temporal aggregation, a three-step process is needed as follows: (i) identify the periods of time in which all values are constant, (ii) compute the universal quantification over these periods, and finally (iii) coalesce the result.

Four cases arise depending on whether the tables WorksOn, Affiliation, and Controls are temporal or not.

Case 1. Only WorksOn is temporal.



The above diagram shows possible values in the three tables and the result of the query. In the diagram W1 and W2 represent two rows of WorksOn relating employee E with projects P1 and P2. At the right of the diagram it is shown that employee E works in department D which controls both projects. Finally, Result shows the periods for which the universal quantification must be calculated, and whether the answer for the period is positive or negative.

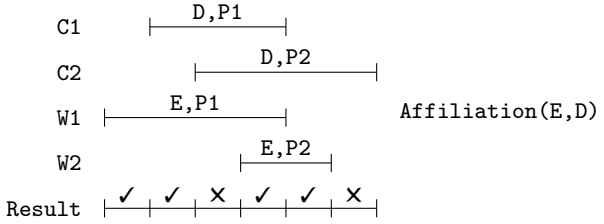
In this case the query can be written in two steps. The first step is shown next.

```
CREATE VIEW TempUnivC1(SSN,FromDate,ToDate) as
  SELECT DISTINCT W1.SSN,W1.FromDate,W2.ToDate
  FROM WorksOn W1, WorksOn W2, Affiliation A
  WHERE W1.SSN=W2.SSN AND W1.SSN=A.SSN
  AND W1.FromDate<W2.ToDate
  AND NOT EXISTS ( SELECT * FROM Controls C
    WHERE A.DNumber=C.DNumber
    AND NOT EXISTS ( SELECT * FROM WorksOn W
      WHERE C.PNumber=W.PNumber AND A.SSN=W.SSN
      AND W.FromDate<=W1.FromDate
      AND W2.ToDate<=W.ToDate ) )
```

The above view looks for two rows (possibly the same) in WorksOn from which the period in the result can be constructed, as well as a row in Affiliation determining the department to which that employee is affiliated. The two inner NOT EXISTS ensure that there is no project controlled by that department in which the employee does not work.

In the second step, the above view must be coalesced.

Case 2. Only Controls and WorksOn are temporal.



The above diagram shows possible values in the three tables and the result of the query. In the diagram C1 and C2 represent two rows of Controls relating department D with projects P1 and P2, while W1 and W2 represent two rows of WorksOn relating employee E with projects P1 and P2. At the right of the diagram it is shown that employee E is affiliated to department D. Finally, Result shows the periods for which the universal quantification must be calculated. Notice that employees may work in projects controlled by departments different to the one to which they are affiliated.

The first step constructs the periods on which the universal quantifier must be computed.

```
CREATE VIEW ProjChangesC2(SSN,Day) as
  SELECT DISTINCT SSN,FromDate
    FROM Affiliation A, Controls C
    WHERE A.DNumber=C.DNumber UNION
  SELECT DISTINCT SSN,ToDate
    FROM Affiliation A, Controls C
    WHERE A.DNumber=C.DNumber UNION
  SELECT DISTINCT SSN,FromDate FROM WorksOn UNION
  SELECT SSN,ToDate FROM WorksOn
CREATE VIEW ProjPeriodsC2(SSN,FromDate,ToDate) as
  SELECT P1.SSN,P1.Day,P2.Day
  FROM ProjChangesC2 P1, ProjChangesC2 P2
  WHERE P1.SSN=P2.SSN AND P1.Day<P2.Day
  AND NOT EXISTS ( SELECT * FROM ProjChangesC2 P3
    WHERE P1.SSN=P3.SSN AND P1.Day<P3.Day
    AND P3.Day<P2.Day )
```

View ProjChangesC2 extracts for each employe the days in which his/her department starts or finishes to control a project, as well as days in which he/she starts or finishes to work in a project. View ProjPeriodsC2 constructs the periods from the above days.

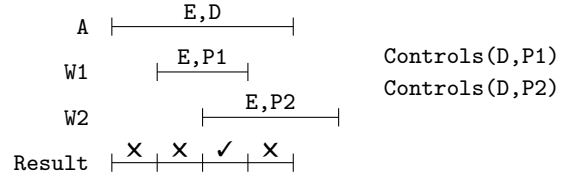
The second step computes the universal quantifier on these periods.

```
CREATE VIEW TempUnivC2(SSN,FromDate,ToDate) as
  SELECT DISTINCT P.SSN,P.FromDate,P.ToDate
  FROM ProjPeriodsC2 P, Affiliation A
```

```
WHERE P.SSN=A.SSN AND NOT EXISTS (
  SELECT * FROM Controls C WHERE A.DNumber=C.DNumber
  AND C.FromDate<=P.FromDate AND P.ToDate<=C.ToDate
  AND NOT EXISTS ( SELECT * FROM WorksOn W
    WHERE C.PNumber=W.PNumber AND P.SSN=W.SSN
    AND W.FromDate<=P.FromDate
    AND P.ToDate<=W.ToDate ) )
```

Finally, the third step is to coalesce the above view.

Case 3. Only `Affiliation` and `WorksOn` are temporal.



The above diagram shows possible values in the tables and the result of the query. In the diagram `W1` and `W2` represent two rows of `WorksOn` relating employee `E` with projects `P1` and `P2`, while `A` represents a row of `Affiliation` relating employee `E` with department `D`. The right of the diagram shows that department `D` controls both projects `P1` and `P2`. Finally, `Result` shows the periods for which the universal quantification must be calculated. As shown in the diagram, no hypothesis is made about the projects in which employees work, i.e., employees may work in projects controlled by departments different to the one to which they are affiliated.

For this query, the first step constructs the periods on which the universal quantifier must be computed.

```
CREATE VIEW Aff_WO(SSN,DNumber,FromDate,ToDate) as
  SELECT DISTINCT A.SSN,A.DNumber,
    maxDate(A.FromDate,W.FromDate),
    minDate(A.ToDate,W.ToDate)
  FROM Affiliation A, WorksOn W WHERE A.SSN=W.SSN
  AND maxDate(A.FromDate,W.FromDate) <
      minDate(A.ToDate,W.ToDate)
CREATE VIEW ProjChangesC3(SSN,DNumber,Day) as
  SELECT DISTINCT SSN,DNumber,FromDate
    FROM Aff_WO UNION
  SELECT DISTINCT SSN,DNumber,ToDate
    FROM Aff_WO UNION
  SELECT SSN,DNumber,FromDate
    FROM Affiliation UNION
  SELECT SSN,DNumber,ToDate FROM Affiliation
CREATE VIEW ProjPeriodsC3(SSN,DNumber,
    FromDate,ToDate) as
  SELECT P1.SSN,P1.DNumber,P1.Day,P2.Day
  FROM ProjChangesC3 P1, ProjChangesC3 P2
  WHERE P1.SSN=P2.SSN
  AND P1.DNumber=P2.DNumber AND P1.Day<P2.Day
  AND NOT EXISTS ( SELECT * FROM ProjChangesC3 P3
    WHERE P1.SSN=P3.SSN AND P1.DNumber=P3.DNumber
    AND P1.Day<P3.Day AND P3.Day<P2.Day )
```

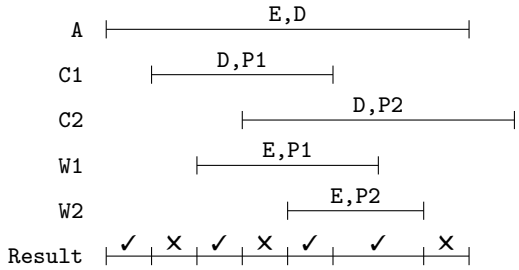
View `Aff_WO` realizes a temporal join of `Affiliation` and `WorksOn` (without the `PNumber` attribute). This collects the days when an employee starts or finishes to work in a project of his/her department. Then, view `ProjChangesC3` extracts those days from the previous view as well as the start and end day of affiliation of an employee to a department. The latter are needed to take into account the period between the beginning of an affiliation and the first time that the employee works in a project (as shown in the diagram), and the period between the last time that the employee works in a project and the end of the affiliation. Finally, view `ProjPeriodsC3` constructs the periods from the above days.

The second step computes the universal quantifier on these periods.

```
CREATE VIEW TempUnivC3(SSN,FromDate,ToDate) as
  SELECT DISTINCT P.SSN,P.FromDate,P.ToDate
  FROM ProjPeriodsC3 P WHERE NOT EXISTS (
    SELECT * FROM Controls C WHERE P.DNumber=C.DNumber
    AND NOT EXISTS ( SELECT * FROM WorksOn W
      WHERE C.PNumber=W.PNumber AND P.SSN=W.SSN
      AND W.FromDate<=P.FromDate
      AND P.ToDate<=W.ToDate ) )
```

Finally, the third step is to coalesce the above view.

Case 4. All tables are temporal.



The above diagram shows possible values in the three tables and the result of the query. In the diagram `W1` and `W2` represent two rows of `WorksOn` relating employee `E` with projects `P1` and `P2`, `C1` and `C2` represent two rows of `Controls` relating department `D` with the two projects, and `A` represents a row of `Affiliation` relating employee `E` with department `D`. Finally, `Result` shows the periods for which the universal quantification must be calculated. Notice that the end date of `W1` does not induce a period in the result since at that time project `P1` is not controlled by `E`'s department.

For this query the first step constructs the periods on which the universal quantifier must be computed as follows.

```
CREATE VIEW Aff_Cont(SSN,DNumber,PNumber,
    FromDate,ToDate) as
  SELECT DISTINCT A.SSN,A.DNumber,C.PNumber,
    maxDate(A.FromDate,C.FromDate),
    minDate(A.ToDate,C.ToDate)
  FROM Affiliation A, Controls C
  WHERE A.DNumber=C.DNumber
  AND maxDate(A.FromDate,C.FromDate) <
      minDate(A.ToDate,C.ToDate)
CREATE VIEW Aff_Cont_WO(SSN,DNumber,PNumber,
    FromDate,ToDate) as
  SELECT DISTINCT A.SSN,A.DNumber,W.PNumber,
    maxDate(A.FromDate,W.FromDate),
    minDate(A.ToDate,W.ToDate)
  FROM Aff_Cont A, WorksOn W
  WHERE A.PNumber=W.PNumber AND A.SSN=W.SSN
  AND maxDate(A.FromDate,W.FromDate) <
      minDate(A.ToDate,W.ToDate)
CREATE VIEW ProjChangesC4(SSN,DNumber,Day) as
  SELECT DISTINCT SSN,DNumber,FromDate
```

```
    FROM Aff_Cont UNION
  SELECT DISTINCT SSN,DNumber,ToDate
    FROM Aff_Cont UNION
  SELECT DISTINCT SSN,DNumber,FromDate
    FROM Aff_Cont_WO UNION
  SELECT DISTINCT SSN,DNumber,ToDate
    FROM Aff_Cont_WO UNION
  SELECT SSN,DNumber,FromDate
    FROM Affiliation UNION
  SELECT SSN,DNumber,ToDate FROM Affiliation
CREATE VIEW ProjPeriodsC4(SSN,DNumber,
    FromDate,ToDate) as
  SELECT P1.SSN,P1.DNumber,P1.Day,P2.Day
  FROM ProjChangesC4 P1, ProjChangesC4 P2
  WHERE P1.SSN=P2.SSN AND P1.DNumber=P2.DNumber
  AND P1.Day<P2.Day AND NOT EXISTS (
    SELECT * FROM ProjChangesC4 P3
    WHERE P1.SSN=P3.SSN AND P1.DNumber=P3.DNumber
    AND P1.Day<P3.Day AND P3.Day<P2.Day )
```

View `Aff_Cont` realizes a temporal join of tables `Affiliation` and `Controls`. This view is then used on view `Aff_Cont_WO` to realize a temporal join of the three tables `Affiliation`, `Controls`, and `WorksOn`. This computes the days in which an employee starts or finishes to work in a project of his/her department. Then, view `ProjChangesC4` extracts those days from the previous views as well as the start and end day of affiliation of an employee to a department. Finally, view `ProjPeriodsC4` constructs the periods from these days.

The second step computes the universal quantifier on these periods as follows.

```
CREATE VIEW TempUnivC4(SSN,FromDate,ToDate) as
  SELECT DISTINCT P.SSN,P.FromDate,P.ToDate
  FROM ProjPeriodsC4 P WHERE NOT EXISTS (
    SELECT * FROM Controls C WHERE P.DNumber=C.DNumber
    AND C.FromDate<=P.FromDate AND P.ToDate<=C.ToDate
    AND NOT EXISTS ( SELECT * FROM WorksOn W
      WHERE C.PNumber=W.PNumber AND P.SSN=W.SSN
      AND W.FromDate<=P.FromDate
      AND P.ToDate<=W.ToDate ) )
```

Finally, in the third step the above view is coalesced.

5. CONCLUSION

In this paper we adopted a practitioner's approach and showed how to compute temporal aggregation and temporal universal quantification in standard SQL. This provides a solution for users that need such time-varying facilities in their applications.

We randomly generated a set of coalesced tables having 100, 1K, 10K, 100K, and 1M 1ines using SQL Server 2000 on a Pentium 4 machine with 1G of RAM. Our experimental results showed that the complexity of the queries is high. Performance may be significantly improved by using procedural SQL (e.g., T-SQL for SQL Server) with cursors in some steps. In particular, the views `SalPeriods`, `SalPeriodsDep`, and the 3 views `ProjPeriodsC2-C4` are extremely inefficient while being conceptually very simple: they just construct time periods from a set of dates. The reason comes from the inner `NOT EXISTS` used for ensuring that `P2.Day` is the next date from `P1.Day`. Replacing these views by T-SQL procedures accessing the dates in ascending order using cursors decreased considerably the complexity. Similarly, for large tables it is more efficient to realize coalescing using cursors. instead of the declarative query given in Section 2.2. Pursuing these optimization efforts constitutes a direction for our future work.

Obviously, the best solution would be that the DBMS provide such time-varying facilities in a native way, since that would increase both database performance and application development productivity. Several solutions have been proposed for computing temporal aggregates such as aggregation trees [5], SB-trees [6] or balanced trees [12].

6. REFERENCES

[1] M. Böhlen, R. Snodgrass, and M. Soo. Coalescing in temporal databases. In *Proc. of the 22th Int. Conf. on Very Large Data Bases*, pages 180–191, 1996.

[2] H. Darwen. Valid time and transaction time proposals: Language design aspects. In Etzion et al. [4], pages 195–210.

[3] C. Date, H. Darwen, and N. Lorentzos. *Temporal Data and the Relational Model*. Morgan Kaufmann, 2002.

[4] O. Etzion, S. Jajodia, and S. Sripada, editors. *Temporal Databases: Research and Practice*. LNCS 1399. Springer-Verlag, 1998.

[5] N. Kline and R. Snodgrass. Computing temporal aggregates. In *Proc. of the 11th Int. Conf. on Data Engineering*, pages 222–231, 1995.

[6] B. Moon, I. Vega-López, and V. Immanuel. Efficient algorithms for large-scale temporal aggregation. *IEEE Trans. on Knowledge and Data Engineering*, 15(3):744–759, 2003.

[7] R. Snodgrass, editor. *The TSQL2 Temporal Query Language*. Kluwer, 1995.

[8] R. Snodgrass. *Developing Time-Oriented Database Applications in SQL*. Morgan Kaufmann, 2000.

[9] R. Snodgrass, M. Böhlen, C. Jensen, and N. Kline. Adding valid time to SQL/Temporal. ANSI X3H2-96-501r2, ISO/IEC JTC1/SC21/WG3 DBL MAD-146r2, 1996.

[10] R. Snodgrass, M. Böhlen, C. Jensen, and A. Steiner. Adding transaction time to SQL/Temporal: Temporal change proposal. ANSI X3H2-96-152r, ISO-ANSI SQL/ISO/IECJTC1/SC21/WG3 DBL MCI-143, 1996.

[11] R. Snodgrass, M. Böhlen, C. Jensen, and A. Steiner. Transitioning temporal support in TSQL2 to SQL3. In Etzion et al. [4], pages 150–194.

[12] J. Yang and J. Widom. Incremental computation and maintenance of temporal aggregates. *Very Large Data Bases Journal*, 12(3):262–283, 2003.

Acknowledgements The author would like to thank Jef Wijsen and Alain Pirotte for their insightful comments that allowed to improve the quality of the paper.

Developing Scientific Workflows from Heterogeneous Services

A. Tsalgatidou, G. Athanasopoulos, M. Pantazoglou
Dept. of Informatics & Telecom.
University of Athens (NKUA)
Athens 15784, Greece
{atsalga, gathanas, michaelp} @di.uoa.gr

C. Pautasso, T. Heinis
Department of Computer Science
Swiss Federal Institute of Technology
ETH Zentrum, 8092 Zurich, Switzerland
{pautasso, heinist} @inf.ethz.ch

R. Grønmo, Hjørdis Hoff, Arne-Jørgen Berre
SINTEF Information and Communication Technology
P.O.Box 124 Blindern, N-0314 Oslo, Norway
{roy.gronmo, hjordis.hoff, arne.j.berre}@sintef.no

M. Glittum
Locus S.A. (LOCUS)
Leif Weldingsvei 6-8, Sandefjord, Norway
mg@locus.no

S. Topouzidou
Athens Technology Center (ATC)
Rizariou 10, Halandri, Athens, Greece
s.topouzidou@atc.gr

ABSTRACT

Scientific WorkFlows (SWFs) need to utilize components and applications in order to satisfy the requirements of specific workflow tasks. Technology trends in software development signify a move from component-based to service-oriented approach, therefore SWF will inevitably need appropriate tools to discover and integrate heterogeneous services. In this paper we present the SODIUM platform consisting of a set of languages and tools as well as related middleware, for the development and execution of scientific workflows composed of heterogeneous services.

1. INTRODUCTION

Scientific WorkFlows (SWFs) have emerged as a response to problems encompassing scientific problem solving techniques and workflow features [4]. The main characteristic of SWFs is the need for integration of heterogeneous systems and components that provide specific functionalities or data. This integration is not an easy task and it is one of the main research issues in this area.

Current trends in software engineering signify a move from component-based to service-oriented development, therefore, we believe that the development of SWFs will be benefited by a service-oriented approach.

There are many types of services, including Web, Peer-to-Peer (P2P) and Grid services, which employ different/incompatible architectural models, protocols, and standards for service description, discovery and composition. However, to our best of knowledge, there is no infrastructure or tools available for facilitating the integration and interoperability of such services.

In this paper we present a platform called SODIUM (Service Oriented Development In a Unified fraMework) which provides a set of languages, tools and corresponding middleware, for modeling and executing SWFs composed of heterogeneous services (Web, P2P and Grid services).

In a service-oriented approach, the various workflow tasks can be executed by various types of services (rather than being programmed from scratch). Initially, these services may not be known. Therefore, following a top-down approach for developing a SWF, we need to model requirements for services which will satisfy specific workflow tasks. SODIUM provides a Visual Composition Language (VSCL) and an associated VSCL Editor which support this modeling task. The following step is to search for appropriate services which can satisfy the requirements of each task. There exist a large number of heterogeneous services with incompatible protocols and standards which makes their discovery a cumbersome task. SODIUM provides a Unified Service Query Language (USQL) and an associated query engine that support the discovery of heterogeneous services in a unified way. Both semantic and Quality-of-Service (QoS) information are utilized to improve the discovery. However, the USQL language and its associated engine are not responsible for maintaining such information; rather, they rely on existing service descriptions (e.g. OWL-S, WSDL-S, WS-QoS, etc.), which are maintained by service providers and are published in the various registries or networks. Selected services substitute the requirements in each workflow task resulting in a concrete SWF model. Next, VSCL graphs are mapped to USCL (Unified Service Composition Language) descriptions which are executed by the SODIUM workflow execution engine. The main purpose of the latter is to

provide an efficient, reliable and scalable platform for executing SWFs composed of heterogeneous services.

In the next section, we present a motivating scenario that illustrates the need for the integration of heterogeneous services in SWFs. Section 3 depicts the overall architecture of the SODIUM platform and the constituent components. Section 4 compares our work with existing related approaches and finally Section 5 gives concluding remarks.

2. MOTIVATING SCENARIO

Our motivating scenario is from the Crisis Management area, where an important task is to determine how to get to a crisis location as fast as possible. For example in case of an accident with severely injured people, it is critical to reach these persons with the appropriate equipment within minutes. In such cases, if the injury causes lack of oxygen to the brain for 3-5 minutes, brain cells start to die and after approx. 15 minutes there are permanent damages. Thus, it is vital that properly equipped ambulances and other rescue units are within a 15-minutes range at all times and places.

This requirement is very difficult to be satisfied due to the vast set of parameters such as accident/injury probability, population density and composition, accessibility, time of day/week/month/year, weather conditions, hospital locations and many others, which need to be considered.

A SWF providing for this scenario could take advantage of information and resources offered by existing or emerging services. Such services may be:

- Web services providing weather information such as temperature and precipitation or traffic conditions from roadside speed sensors and video surveillance cameras.
- Grid services providing driving route calculations, historical incident information and "response range" calculations based on current positions and conditions.
- P2P services providing information about the locations and status of emergency vehicles and messaging facilities to the emergency vehicles with reposition message commands.

It is therefore imperative that a SWF supporting this scenario is able to integrate heterogeneous services such as the ones mentioned above.

3. SODIUM Architecture

Figure 1 provides an overview representation of the SODIUM platform. As we can see, SODIUM introduces a let of languages, tools and associated middleware.

The languages introduced by the SODIUM platform are:

- A *Visual Service Composition Language (VSCL)* for designing workflow graphs at multiple levels of details.
- A *Unified Service Composition Language (USCL)* to facilitate the construction of executable workflows composing of and invoking various types of services.
- A *Unified Service Query Language (USQL)* to cater for the open and unified discovery of heterogeneous services enabling the preservation of the autonomy of service registries.

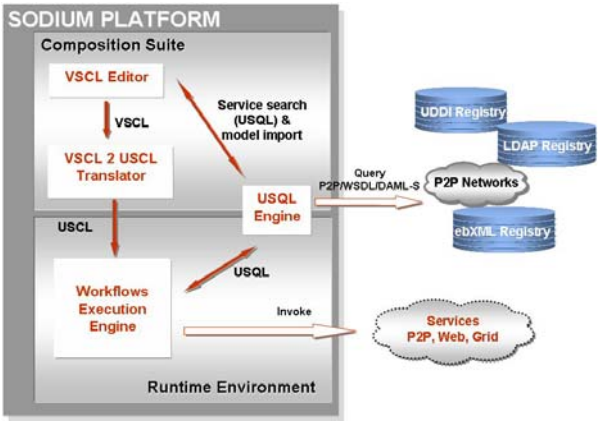


Figure 1: SODIUM Platform overview architecture

With respect to tools and middleware, the SODIUM platform provides the following:

- A Visual Service Composition Suite comprising:
  - A *Visual Editor* enabling the construction and analysis of VSCL Graphs.
  - A Translation mechanism enabling the transformation of the VSCL graphs into USCL.
- A Run Time Environment comprising components necessary for the execution of the composite services:
  - A search engine, namely *USQL Engine*, which submits queries to heterogeneous service registries, utilizing USQL.
  - A *Workflow Execution Engine* which executes workflows written in USCL. The workflow engine invokes the different types of services and/or submits USQL queries to the USQL Engine and subsequently invokes the returned services.

In the following, we describe the three main components of the SODIUM platform, i.e. the Visual Editor, the USQL Engine and the Workflow Execution Engine.

3.1 Visual Editor

The Visual Editor is the SODIUM tool for creating SWF models as VSCL graphs. The first step in constructing VSCL graphs is to break down the workflow into tasks, which can interoperate in order to finally achieve the overall goal. This workflow model of tasks is called an *abstract model* since there are no selected concrete services identified in this phase. This abstract model is used to search for appropriate candidate services to realize each of the abstract tasks. When chosen services are selected for each abstract task, the result is a *concrete model*. VSCL is based on the Unified Modeling Language (UML) [22].

There are extensions to handle Web services, P2P services, Grid services, semantics, QoS, and the relationship between the abstract and the concrete model.

Figure 2 shows an abstract workflow model based on the crisis management scenario presented in section 2. It is abstract since we have only identified the tasks without any concrete service implementations. Each task is defined with enough semantic and QoS information so as to enable a proper search for candidate services. Each task is represented as a UML activity with a stereotype indicating the type of service we are looking for. The goal is to create a service-oriented workflow that monitors the position of ambulance vehicles and sends messages to the ambulances so that they can reposition themselves to achieve a better coverage in a given area. The simplified scenario proposes an ideal ambulance coverage to mean at least one ambulance with a maximum 15-minutes response range for any given position within the area. More realistic scenarios would take into account issues like population density and accident frequency that require better ambulance coverage in certain areas.

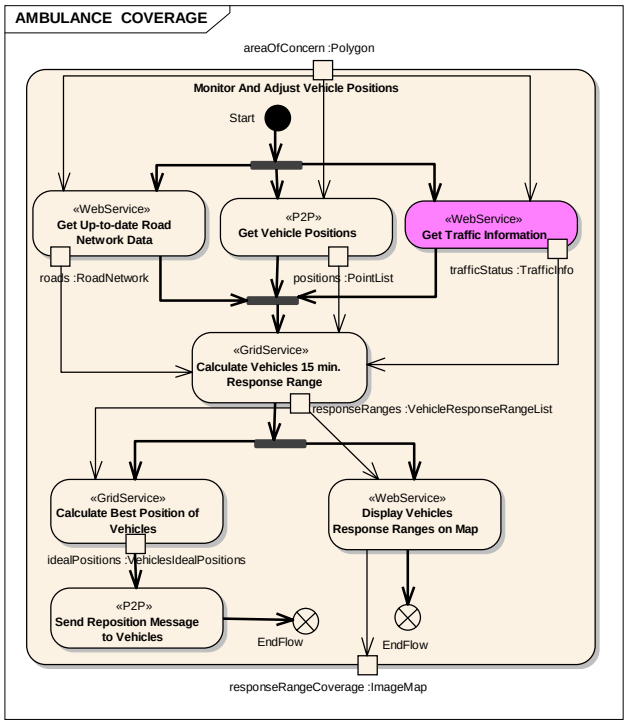


Figure 2: Workflow with heterogeneous services

The workflow takes as input an *areaOfConcern* (i.e. geographical region of concern) which is dispatched at the same time to three different services running in parallel: (1) a Web service returning up-to-date road network data, (2) a P2P service returning the current position of all ambulance vehicles and (3) a Web service returning current traffic information. Results of the three services are forwarded to a Grid service, which calculates the 15-minutes response range for each vehicle. This information is sent to a Web service used to display it on a map covering the original *areaOfConcern* and to a Grid service to calculate the ideal repositioning of the vehicles. These positions are sent to each vehicle by invoking a P2P service. The map produced by the Web service is the output data object returned by the workflow.

All tasks in the abstract workflow model are associated with Quality-of-Service (QoS) requirements and semantically grounded definitions. This improves the precision of the query for concrete services since inappropriate services are omitted from the search results.

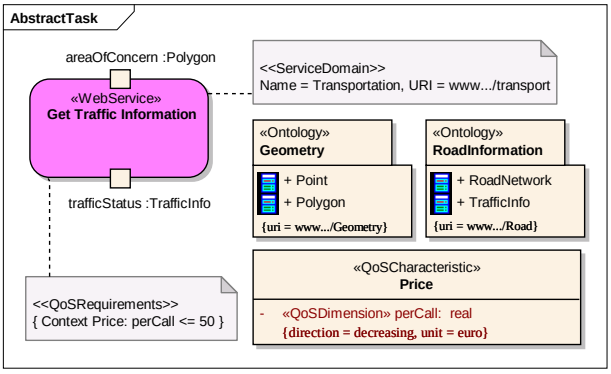


Figure 3: Detailed task with QoS requirement and semantics

Figure 3 illustrates how we can model QoS and semantics for the *Get Traffic Information* task of Figure 2. A note is attached to the task with a pre-defined stereotype *QoSRequirement*. The QoS requirement specifies that the 'price per call' of the service must be less-than-or-equal-to 50 euro. The 'price per call' is a QoS concept that needs to be defined within a UML class with the stereotype *QoSCharacteristic*. The *Get Traffic Information* task is also semantically annotated by defining a service domain and by defining the input and output data objects as semantic types. The semantic types of the data objects are placed in a UML package stereotyped as *Ontology*. The ontology package corresponds to an existing domain ontology identified by the UML tagged value *uri*. The *areaOfConcern* input data object is assigned the *Polygon* type which is a semantic type defined within the *Geometry* ontology, represented by a UML package. It should be noted, that any domain ontology could be practically used as the source for semantically annotating tasks. VSCL, along with USQL, are flexible enough so as to accommodate different ontologies. Nevertheless, an upper domain ontology has been established and is maintained by the USQL Engine, within the scope of SODIUM. More details regarding the upper ontology are provided in the next section (3.2), as well as in [23].

The semantic modeling of services is an extension to UML provided by the SODIUM project. For the QoS notation we

recommend the use of the OMG's UML profile for modeling QoS and Fault Tolerance [14].

A concrete workflow model extends the abstract workflow model with chosen services (that have been discovered via the USQL Engine) for each of the abstract tasks. The concrete workflow model can then be automatically translated into the lexical USCL language which can be executed by the workflow execution engine [24]. The next section explains how the USQL engine is used to discover appropriate services to register in the concrete workflow model.

3.2 USQL Engine

The *USQL Engine* is used for discovering services by searching heterogeneous service registries. The engine implements *USQL (Unified Service Query Language)*, an XML-based query language providing all necessary structures and elements to cater for the unified and standards-based service requests over heterogeneous registries and/or networks. USQL allows requestors to formulate expressive, semantically enhanced queries, which reflect their actual needs and requirements. Queries may also be enriched with the specification of QoS criteria, so as to bring the resulting services as close as possible to the demands of real-world scientific applications.

An example USQL request reflecting the service requirements of Figure 3 is depicted in Figure 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<USQL version="1.0">
  <find_servicesRequest>
    <Where>
      <Service serviceType="WebService">
        <ServiceDomain ontologyURI="...">Transportation</ServiceDomain>
        <ServiceQoS>
          <ServicePrice currency="EUR" context="perCall" valueIs="equalOrLess">
            50
          </ServicePrice>
        </ServiceQoS>
        <Operation>
          <OperationName>GetTrafficInformation</OperationName>
          <Input>
            <name>areaOfConcern</name>
            <semantics ontologyURI="...">Polygon</semantics>
          </Input>
          <Output>
            <name>trafficStatus</name>
            <semantics ontologyURI="...">TrafficInfo</semantics>
          </Output>
        </Operation>
      </Service>
    </Where>
  </find_servicesRequest>
</USQL>
```

Figure 4: A sample USQL request

The main concept underlying the USQL Engine framework is the abstraction regarding registry details, from the requestor's perspective. This is achieved with the adoption of a domain-centric categorization of the various supported registries, depending on the service advertisements they host. Domain information provided by the requestor is exploited by the engine so as to identify, access and query the appropriate registries in a transparent manner.

The USQL Engine accomplishes this type of domain-driven registry categorization with the introduction of an *Upper Ontology*, which provides a set of classes along with their properties and relations, thus allowing for the application of reasoning within the course of service discovery. The upper ontology associates registries with domains, and concepts with domains. Concepts may be either operations or data that are relative to a specific domain, and are used for semantically annotating USQL requests, as well as in the matchmaking process. Moreover, an ontology mapping mechanism is used to ensure interoperability, as far as support for service descriptions abiding by different ontology frameworks and/or vocabularies is concerned. Maintenance of the Upper Ontology is part of the USQL Engine configuration process, so that service requestors can focus on the thorough and consistent expression of their queries.

The USQL engine architecture is distinguished by its high degree of openness and extensibility, which is achieved by using plug-in mechanisms in order to accommodate virtually any type of service, registry, as well as their governing protocols and standards. The plug-ins used for this purpose can be integrated in a flexible manner, so as to enable different configurations and to broaden the range of supported registries.

The USQL Engine is a crucial component within the context of a scientific workflow engine facilitating the discovery of heterogeneous services that are used for solving a scientific problem. The service discovery results are used to transform abstract workflow graphs conveying orchestrated tasks and their respective requirements into concrete service workflows that are then executed by the SODIUM workflow engine. Alternatively, the USQL Engine may be used at run-time for the discovery of appropriate services to fulfill specific tasks within the workflow.

3.3 Workflow Execution Engine

The engine receives USCL documents containing the definitions of the workflows and exposes an API for initiating, monitoring, and managing their execution. In addition, workflows can be exposed as Web services [10]; hence, it is possible to access them from client applications through standard interfaces.

The core infrastructure used to run USCL workflows is described as follows. The execution of a workflow begins with a request sent through the corresponding API of the engine. The engine API queues the request and handles it by creating and enacting a new workflow instance. The current state of the execution of a process is used to determine which tasks should be invoked based on the control and data flow dependencies that are triggered by the completion of the previous tasks.

Invocation of the tasks equals to invocation of their respective services and is achieved with the appropriate plug-ins. These provide a concrete implementation of the service invocation mechanisms and protocols. After the

execution of the task has been finished, the state of the corresponding process is updated and the execution of the workflow continues. Execution of the workflow is separated from the execution of its tasks since these operations have a different level of granularity and quite often the execution of a task may last significantly longer than the time taken for scheduling it. Thus, the engine supports parallel invocation of multiple tasks belonging to the same process. Furthermore, a slow task does not affect the execution of other processes running concurrently because these two operations are handled asynchronously by different threads.

The SODIUM workflow engine provides for the persistence of the state information about the process instances following a design that has been influenced by many requirements, such as performance, reliability, and portability across different data repositories. Access to the state information of the workflow instances is provided in terms of a key-value data model which uniquely identifies a certain data (or meta-data) value associated with a process (and task) instance. The state information data model is independent of the physical location of the data, so that it is possible to use caching to exploit locality and – for increased availability – replicate some of the values. Along these lines, in order to provide a level of scalability which complies with what is required in a scientific workflow setting, state management can be optimized to keep only a subset of all of the workflow instances in memory and, for instance, swap workflows whose execution has been completed to secondary storage, in a so-called process history space. In this way, the SODIUM engine gives access to the state of past executions to enable workflow profiling and optimization, caching of already computed results as well as lineage tracking analysis. Additionally, this functionality can be implemented with several different storage technologies along the full spectrum of the persistence versus performance overhead trade off. In the context of the crisis management example, this functionality helps to optimize the performance of the workflow execution as follows. The expensive re-computation of the 15-minutes response range can be avoided if neither the vehicle positions, nor the traffic conditions have changed with respect to a previous execution. Similarly, it is possible to avoid the potentially large download of the same road network data for repeated executions of the workflow with the same input *areaOfConcern*.

The architecture of the SODIUM workflow execution engine employs plug-ins to support an open set of heterogeneous service invocation mechanisms.

3.4 Web Service Plug-in

The first of such plug-ins is responsible for the invocation of standards-compliant Web services described by WSDL [2]. These services are remotely accessible through the SOAP protocol [1]. By utilizing the information returned by a USQL response[1] which provides the URI of a WSDL document describing the selected service, the port, service and operation name as well as the arguments for the operation, the Web service plug-in invokes the Web service by dynamically assembling a SOAP message and sending it to the service provider. Upon receiving a response, the results are extracted and passed back to the workflow engine. For example, in order to invoke the "Get Traffic Information" Web service of Figure 3, the plug-in uses the URI to the WSDL returned by the USQL response to the USQL query of Figure 4, selects the port bound to the SOAP protocol and sends a request message to the *GetTrafficInformation* operation. This message contains the XML serialization of the input parameter *areaOfConcern*.

3.5 Grid Service Plug-in

The Grid service plug-in has been developed in accordance to the Web Services Resource Framework (WSRF) [3] specifications, which have been defined as to shift from the rather stateless paradigm of Web services to the stateful model of Grid services. To do so, WSRF loosely couples a Web service with a stateful resource and provides well-defined methods to access and manage its state.

Thus, in contrast to using only the service URI as is the case for Web services, Grid services also require a resource instance identifier. Therefore, in addition to the arguments passed to the Web service plug-in, the Grid service plug-in also requires a resource instance identifier.

3.6 P2P Service Plug-in

As far as P2P services are concerned, JXTA [19] is one of the most well known platforms for the development of P2P service-oriented applications, therefore JXTA services have been appointed as a supported type of P2P services, with respect to the SODIUM platform. Nevertheless, the mechanisms and facilities provided by the JXTA platform for the description and invocation of JXTA P2P services are rather limited or vague. To address this, we follow an approach based on the use of enhanced description documents and a library of java classes facilitating the binding and invocation of JXTA P2P services.

The pursued approach does not modify the infrastructure or protocols that are used by the JXTA services; rather, it enhances them so as more advanced discovery and binding mechanisms can be used. Furthermore, the P2P service model is not infringed, since the workflow engine actually

[1] A USQL query is submitted to the USQL engine either at design time (in which case service-related information returned by the USQL response is integrated in the VSCL graph, then mapped to a USCL document and subsequently utilized by the execution engine for invoking the service) or at run time, thus supporting dynamic discovery of services.

becomes a node in the respective P2P network, through the plug-in.

4. RELATED WORK

SODIUM provides for the service-oriented development of SWFs. Its contribution lies in the areas of SWF modeling (VSCL with editor), execution (USCL with execution engine), and in service discovery (USQL with query engine). In the following we compare the SODIUM results with existing work in each of these areas.

Similar to KEPLER [5][12] SODIUM's workflow modeling languages (VSCL and USCL) are primarily based on data flow constructs, as this is the most common representation used to model scientific computations. While KEPLER is confined to Web services, SODIUM allows for the discovery and composition of Grid, as well as P2P services, providing support for semantics and QoS metadata which can be used for optimized service selection [9]. Triana [21] is a framework for composing scientific applications. Unlike USCL, the supported workflow language does not provide explicit support for control constructs, while its visual representation is not standard-based with respect to VSCL. For more information on other research projects (e.g., Askalon, Unicore, Karajan) related to scientific workflow management, we refer the reader to [13] and [25].

Considering the scale of the data and the computational resources required by scientific applications, scientific workflow systems must take into account resource management and scheduling features typical of high performance computing environment and tools [7], [11]. In this regard, the Pegasus workflow mapping system [8] shares with SODIUM the idea of mapping a workflow between different levels of abstraction, where an abstract workflow is dynamically bound with run-time information describing the Grid resources that are used to execute its activities. A different kind of mapping is related to providing data transformation capabilities so that mismatching scientific data sources and incompatible tools can be integrated e.g., [6]. Thanks to its extensibility, SODIUM features a rich set of data transformation techniques such as XSLT [16], XQuery [17] as well as QVT [15] so that the most optimal one in terms of run-time performance and development effort can be applied.

The use of a service-oriented approach to SWF development introduces the need for service discovery which is only partially addressed by other approaches to SWF development [20]. In the areas of Web, Grid, and P2P services, service discovery is performed with the use of custom and incompatible APIs and discovery mechanisms offered by registries and networks [18] [19]. Nevertheless, service-oriented development lacks a query language that would enable accessing and querying heterogeneous registries in a unified and standards-based manner. Moreover, exploitation of semantics and QoS within service descriptions proves to be a crucial part of service discovery. USQL and its engine address these issues and constitute a stepping stone to the unification of the various heterogeneous service areas.

5. DISCUSSION AND CONCLUSIONS

Service Oriented Computing (SOC) is a new trend in software engineering. SOC is already affecting the development of business oriented systems and we believe that it will inevitably affect the development of SWFs turning them into service compositions. However, the heterogeneity in protocols and standards of existing service types is a major obstacle for the discovery of services and their integration in scientific workflows.

In this paper we briefly described a platform called SODIUM which provides tools, languages and related middleware for supporting the whole lifecycle of SWFs (i.e. from requirements modeling to their execution) composed of heterogeneous services. Specifically, SODIUM supports abstract as well as concrete modeling of workflows (by providing the VSCL language and editor), uniform discovery of constituent heterogeneous services (through USQL and the query engine) and execution of service workflows (through the USCL Engine).

The open and extendable architecture of SODIUM doesn't alter the underlying protocols and infrastructure used by the various services, but rather hides the specific details from the workflow developers. Furthermore, besides the service types currently supported, i.e. Web, Grid and P2P, SODIUM provides for the easy integration of any other service type.

Acknowledgement. This work has been partially supported by the European Commission under the contract IST-FP6-004559 (project SODIUM: Service Oriented Development in a Unified fraMework).

6. REFERENCES

[1] W3C, 2001, *SOAP 1.1*, http://www.w3.org/TR/SOAP

[2] W3C, 2001, *WSDL 1.1*, http://www.w3.org/TR/wsdl

[3] OASIS, 2004, *WSRF*, ***http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf***

[4] Singh, M. P., and Vouk, M. A. *"Scientific Workflows: Scientific Workflow Meets Transactional Workflow."* NSF Workshop on Workflow and Process Automation in Information Systems: State of the Art and Future Directions, Athens, GA, USA, 1996

[5] I. Altintas et al, ***Kepler: An Extensible System for Design and Execution of Scientific Workflows***, SSDBM'04, 21-23 June 2004, Santorini Island, Greece.

[6] Shawn Bowers and Bertram Ludäscher ***An Ontology-Driven Framework for Data Transformation in***

Scientific Workflows In Procs of DILS'04, Leipzig, Germany, Springer, LNCS, volume 2994.

[7] J. Cao, S. A. Jarvis, S. Saini, G. R. Nudd, *GridFlow: Workflow Management for Grid Computing*, Proc. of the 3rd International Symposium on Cluster Computing and the Grid, Tokyo, Japan, May 2003.

[8] E. Deelman et al, ***Pegasus : Mapping Scientific Workflows onto the Grid***, *Across Grids Conference 2004*, Nicosia, Cyprus, 2004.

[9] R. Grønmo, M. C. Jaeger *Model-Driven Methodology for Building QoS-Optimised Web Service Compositions*. In Procs of DAIS '05 Athens, Greece.

[10] T. Heinis, C. Pautasso, G. Alonso, O. Deak, ***Publishing Persistent Grid Computations as WS Resources***, In: Proceedings of the 1st IEEE International Conference on e-Science and Grid Computing (e-Science 2005), Melbourne, Australia, December 2005.

[11] S. Krishnan, P. Wagstrom, and G. von Laszewski. ***GSFL: A Workflow Framework for Grid Services***. *Technical Report, Argonne National Laboratory*, Preprint ANL/MCS-P980-0802, August 2002.

[12] B. Ludäscher et al, ***Scientific Workflow Management and the Kepler System***, *Concurrency and Computation: Practice & Experience*, Special Issue on Scientific Workflows, to appear, 2005.

[13] Bertram Ludäscher and Carole Goble, ***Special Section on Scientific Workflows***, SIGMOD Record, 34(3), September 2005.

[14] OMG, *UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms*, OMG Final Adopted Specification, ptc/04-09-01

[15] QVT-Merge Group. *Revised submission for MOF 2.0 Query/Views/Transformations* OMG document: ad/2004-10-04 version 1

[16] XSLT, ***http://www.w3.org/TR/xslt***

[17] XQuery, ***http://www.w3.org/TR/xquery/***

[18] UDDI, ***http://www.uddi.org***

[19] JXTA Technology, ***http://www.jxta.org***

[20] I. Altintas, E. Jaeger, K. Lin, B. Ludaescher, A. Memon, *"A Web Service Composition and Deployment Framework for Scientific Workflows"*, ICWS 2004

[21] Matthew Shields, Ian Taylor, *"Programming Scientific and Distributed Workflow with Triana Services"*, In *Proceedings of Workflow in Grid Systems Workshop in GGF10*, at Berlin, Germany, March 2004

[22] OMG, "*Unified Modeling Language: Superstructure, version 2.0*", OMG Final Adopted Specification, ptc/04-10-02

[23] Aphrodite Tsalgatidou, George Athanasopoulos, Michael Pantazoglou, *"Semantically Enhanced Discovery of Heterogeneous Services"*, 1st International IFIP/WG12.5 Working Conference on Industrial Applications of Semantic Web (IASW2005), 25-27 August 2005, Jyväskylä, Finland

[24] David Skogan, Hjørdis Hoff, Roy Grønmo, Tor Neple, *"D9-Detailed Specification of the SODIUM Service Composition Suite"*, SODIUM (IST-FP6-004559) project's deliverable, June 2005

[25] Jia Yu and Rajkumar Buyya, "*A Taxonomy of Workflow Management Systems for Grid Computing*", Technical Report, GRIDS-TR-2005-1, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, March 10, 2005

Impact of Double-Blind Reviewing on SIGMOD Publication Rates

Samuel Madden
MIT CSAIL
madden@csail.mit.edu

David DeWitt
University of Wisconsin, Madison
dewitt@cs.wisc.edu

Background

Starting with the 2001 SIGMOD conference, the SIGMOD Chair, in consultation with the SIGMOD Advisory Committee, imposed a double blind rule on all future SIGMOD conferences. While there are many reasons why double-blind reviewing might be a good idea, the one most frequently cited is that it is fairer to more junior researchers. It is not, however, without its problems, including anecdotal reports of papers being rejected because their authors failed to cite their own papers as related work in order to not violate the anonymity rules and a complication of the job of the program chair who must interpret and enforce the double-blind rules. One very qualified individual turned down an offer to be PC chair of an upcoming SIGMOD conference because he did not want to have to deal with the headaches of double-blind reviewing.

Now that we have had five years of double-blind SIGMOD conferences (2001-2005) we thought it might be useful to determine whether the use of double blind reviewing has had a significant impact on the rate at which more "senior" researchers have their papers accepted. Since DBLP data is not yet available for SIGMOD 2006 it was not included.

Methodology

Our first step was to define a study group to evaluate the impact of double blind refereeing. As our study group we selected those individuals who have published 20 or more papers in SIGMOD and VLDB conferences. There are 28 such individuals; we filtered out three of these researchers who have not published papers in either of the past two SIGMODs or VLDBs. We will term the remaining 25 researchers "prolific". We factored out demo proposals, panel descriptions, tutorials, and most industrial papers by counting only papers that were 5 pages or longer. We have also studied the results with a slightly more or less inclusive definition of prolific researchers and found the results to be essentially unchanged.

As the basis for comparison we used DBLP publication data from the SIGMOD 1994-2000 conferences and the 1994-2005 VLDB conferences, all of which were not double blind. We decided not to use data from earlier SIGMOD or VLDB conferences in an attempt to minimize the impact that longitudinal variations in productivity would have on our results.

Results

Table 1 summarizes the data we used for our analysis. Here, double blind conferences (SIGMOD 2001 – 2005) are shown in bold and red. The three sets of columns indicate the average number of papers per prolific researcher, the total fraction of papers with at least one prolific researcher as an author, and the total number of papers in both VLDB and SIGMOD for each year.

Tables 1 clearly indicates several phenomena. Like everyone else, our "prolific researchers" have their good years and bad years when it comes to getting their papers accepted. VLDB 2005 was a particularly bad year while SIGMOD 2000 (non-double blind) and 2004 (double blind) were very good years.

The fraction of papers by prolific researchers indicate the same trends as above. Some years (such as VLDB 2005) were not as bad for our prolific researchers as when viewed as the percentage of papers represented in each conference because the total number of papers accepted has risen (VLDB 2005 had more than 100 papers, including the industrial track, while SIGMOD 1999 had less than 50.) This also probably explains the rising trend in number of accepted papers for prolific researchers in VLDB prior to 2005.

The averages of these various statistics are shown in Tables 2 and 3; overall, imposing double blind refereeing has not had significant impact on the publication rate for this group of researchers.

Year	Papers/Famous Person		Total Papers		Fraction Famous Papers	
	SIGMOD	VLDB	SIGMOD	VLDB	SIGMOD	VLDB
1994	0.81	0.73	42	65	0.48	0.28
1995	0.54	0.73	36	59	0.37	0.31
1996	0.88	1.07	47	49	0.47	0.55
1997	0.92	0.85	42	55	0.55	0.38
1998	0.73	0.69	42	52	0.43	0.33
1999	0.88	0.81	42	58	0.53	0.35
2000	1.00	0.88	48	58	0.52	0.38
2001	**0.77**	0.81	**44**	66	**0.44**	0.31
2002	**0.81**	1.15	**50**	91	**0.40**	0.32
2003	**0.85**	1.15	**53**	84	**0.40**	0.34
2004	**1.34**	1.53	**69**	102	**0.49**	0.38
2005	**0.81**	0.92	**66**	103	**0.31**	0.22

Table 1: Publication Statistics by Year Per Conference. Double blind conferences are highlighted in bold and in red. Papers/prolific researcher indicates the average number of papers accepted into the conference per prolific researcher. Fraction prolific represents the total fraction of accepted papers that had at least one prolific researcher as an author.

	SIGMOD	VLDB	Total
01-05	**0.91**	1.11	1.01
94-00	0.82	0.82	0.82
Total	0.86	0.94	0.92

Table 2: Average of number of papers by prolific researchers in double blind and non-double blind conferences. Here, SIGMOD was double blind from 2001-2005, and was not double blind from 1994-2000. VLDB has never been double blind.

	SIGMOD	VLDB	Total
01-05	**0.41**	0.31	0.36
94-00	0.48	0.37	0.42
Total	0.45	0.35	0.40

Table 3: Average fraction of papers by prolific researchers in double blind and non-double blind conferences.

Figures 1, 2, and 3 show graphs of the numbers in Table 1.

Summary

Based on the results in Table 1, it is apparent that double-blind reviewing has had essentially no impact on the publication rates of more senior researchers in the database field. There are two possible takeaways. One is that imposing double-blind reviewing on authors of SIGMOD papers has had no effect while requiring a significant effort for both authors and program chairs. The other is that junior researchers submitting papers to VLDB conferences are not being significantly impacted by its use of a non-double blind review process (since publication rates have been held approximately constant and senior researchers share of papers has not risen).

Though there may be other compelling reasons for maintaining a double-blind reviewing process, such as maintaining a perception of fairness, our analysis shows that the commonly cited benefit of an *actual* increase in fairness does not, in reality, seem to exist.

Finally, we realize that a better way of gauging the impact of double-blind reviewing would be determine the actual acceptance rates for our set of "prolific researchers". Unfortunately, this data is not readily available and "prolific researchers" we contacted could not even generate their own actual acceptance rates.

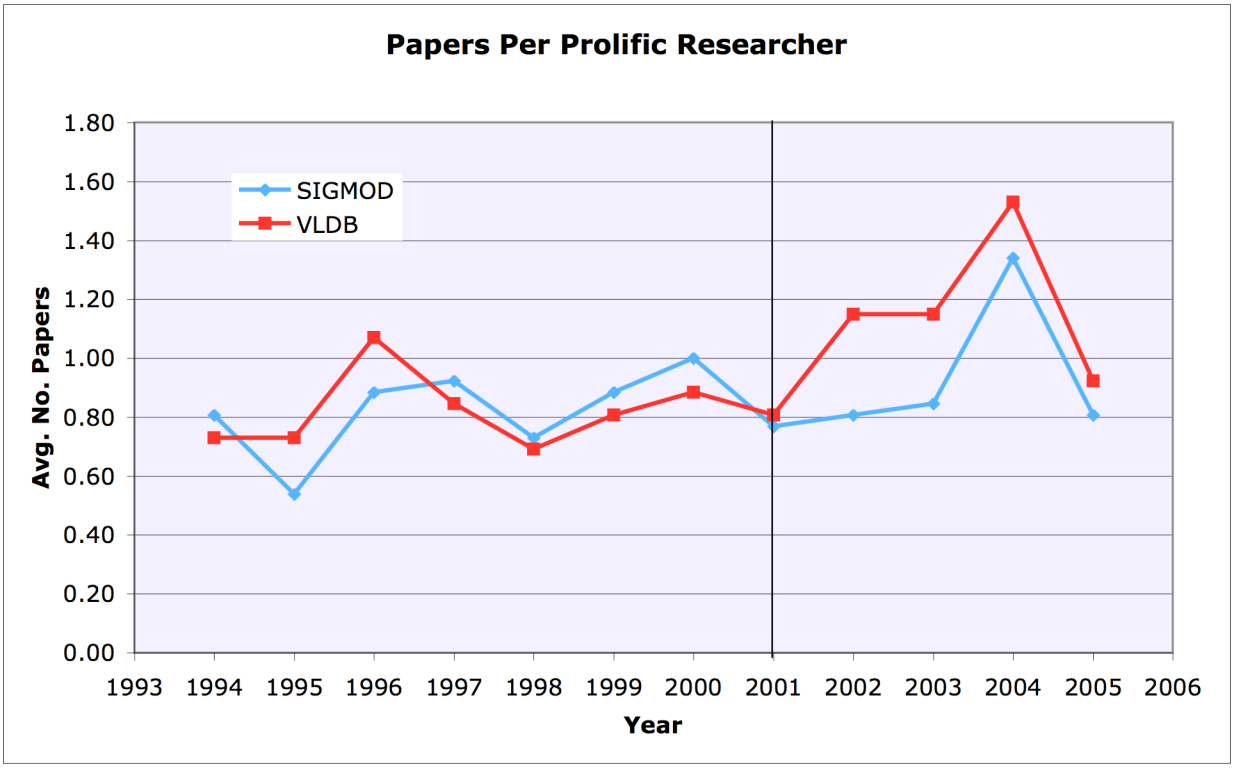


Figure 1: Graph showing average # papers per prolific researcher per conference by year

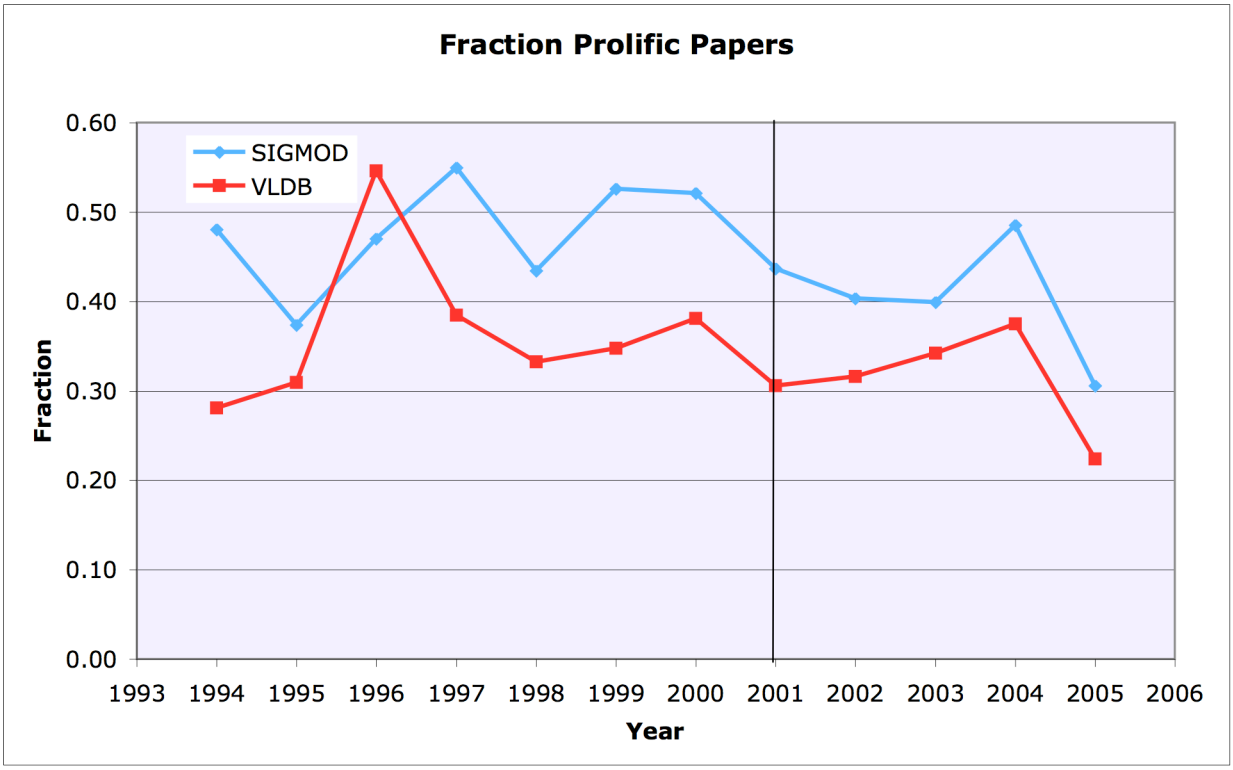


Figure 2: Graph show fraction of papers by prolific researchers per conference per year

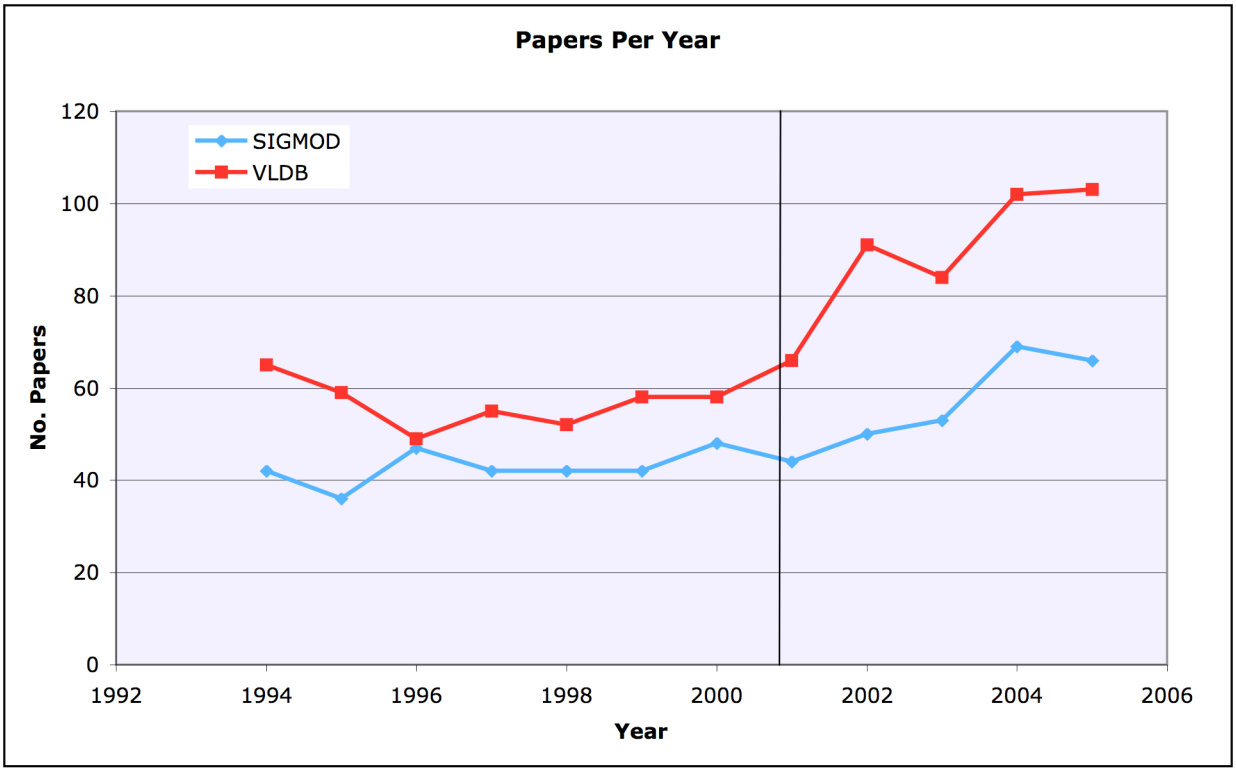


Figure 3: Graph showing total number of papers per conference per year

"A Veritable Bucket of Facts" Origins of the Data Base Management System

Thomas Haigh

University of Wisconsin--Milwaukee
Bolton Hall, Room 582, P.O. Box 413, Milwaukee, WI 53201-0413
thaigh@computer.org

ABSTRACT

The data base concept derives from early military on-line systems, and was not originally associated with the specific technologies of modern data base management systems. While the idea of an integrated data base, or "bucket of facts," spread into corporate data processing and management circles during the early 1960s, it was seldom realized in practice. File-processing packages were among the very first distributed as supported products, but only in the late 1960s were they first called "data base management systems," in large part through the actions of the Data Base Task Group of the Committee on Data Systems Languages (CODASYL). As the DBMS concept spread, the data base itself was effectively redefined as the informational content of a packaged DBMS. Throughout the process, managerial descriptions of the data base as a flexible and integrated repository for all corporate data stood in sharp contrast with the useful but limited nature of actual systems.[1]

1. INTRODUCTION

The Data Base Management System (DBMS) is the foundation of almost every modern business information system. Virtually every administrative process in business, science or government relies on a data base. The rise of the Internet has only accelerated this trend – today a flurry of database transactions powers each content update of a major website, literature search, or internet shopping trip. Yet very little research addresses the history of this vital technology, or that of the ideas behind it. We know little about its technical evolution, and still less about how its usage has changed over time. [2]

A data base management system is a very complex piece of system software. A single DBMS can manage multiple data bases, each one usually consisting of many different tables full of data. The DBMS includes mechanisms for application programs to store, retrieve and modify this data and also allows people to query it interactively to answer specific questions. Specialists, known as Data Base Administrators (DBAs) control the operation of the DBMS and are responsible for the creation of new data bases and the definition of the table structures used to store data. One of the most important features of the DBMS is its ability to shield the people and programs using the data from the details of its physical storage. Because all access to stored data is mediated through the DBMS, a data base can be restructured or moved to a different computer without disrupting the programs written to use it. The DBMS polices access to the stored data, giving access only to tables and records for which a given user has been authorized.

Today, corporate computer staff would usually conceive of a data base as the content of a data base management system. (In fact, the two concepts are so closely associated that DBMSs such as Oracle are often simply called data bases, even by IT specialists). Historically, though, the two ideas were distinct. The data base concept originated around 1960, approximately ten years before the idea of a DBMS gained general currency. The data base concept originated among the well-funded cold war technologists of the military command and control, and so was associated with the enormously complex and expensive technologies of on-line, real-time, interactive computer applications. By the mid-1960s it had entered managerial discourse, and was used to describe the huge pools of shared data needed to construct a "totally integrated management information system" (MIS) to integrate every aspect of the management of a large corporation.

On a technical level, however, the DBMS evolved from a more humble class of programs known as "file management systems", created within the unglamorous world of corporate data processing to simplify the creation of programs for routine administration. The data base management system conflated the managerial concept of the data base with the specific technology of the file management system. As this paper shows, in practice the DBMS worked well as a technical system to aid application programmers, but disappointed as a managerial panacea. Most early DBMS systems were used primarily for routine applications, were not queried directly by managers, and did not support the integration of all corporate data. In addition, while the corporate data base had originally been conceived as a repository of all important managerial information, actual DBMS technology supported only the kind of highly structured regular records with which earlier file management systems had been adept.

The story of the DBMS therefore provides an interesting example of the process by which particular technologies with very specific

[1] This is a revised version of an article originally published in W. Boyd Rayward and Mary Ellen Bowden, eds., The History and Heritage of Scientific and Technological Information Systems: Proceedings of the 2002 Conference (Medford, New Jersey: Published for the American Society for Information Science and Technology and the Chemical Heritage Foundation by Information Today, Inc., 2004.) http://www.chemheritage.org/events/asist2002/proceedings.html

[2] While many data base textbooks include a few pages on the development of data base theory along with their introductory definitions – for example [41] does this well – this can mean little when stripped of its historical context. The closest thing to a detailed history is a quarter-century old technical primer [44, pages 19-29]. A short history, focusing on the role of public funding in the emergence of the relational model, is found in [77, ch. 6]. On the technical side, detailed comparisons of early systems are given in [19, 31, 101, 111].

qualities and distinctive strengths and weaknesses are promoted instead as universal solutions. The same pattern has been seen many times: in early discussion of information retrieval as a problem that could be solved for the general case, with the christening of computers as information technology, and with more recent attempts to sell systems for "data warehousing", "data mining" or "knowledge management" as universally applicable technical solutions to organizational needs. In all these cases, acceptance of the idea of information as a generalized quantity that can be stored in and processed by machines serves to elide the difference between very broad human or managerial concepts of information and the far more constrained capabilities of specific automated systems.

2. THE DATA BASE AND THE MANAGEMENT INFORMATION SYSTEM

During the 1970s, when data base management systems were first promoted to corporate managers, they were sold as the technological means by which all of a company's computerized information could be assimilated into a single integrated pool of data. This idea was not, however, a new one. Indeed, its widespread discussion among experts on the managerial applications of computers dates back to the late 1950s, several years before the term "data base" was used in this context. To understand the initial concept of the data base, and its appeal, we must therefore begin by examining the concept of the Management Information System (MIS).

In March 1960, a senior representative of Arthur D. Little, then the largest and longest established management consulting firm, addressed his colleagues at a conference organized by the American Management Association to discuss new applications of computer technology to the problems of corporate administration [102]. Milton D. Stone was, like many of his fellow speakers, enthusing about the incredible potential of the Management Information System, then a very new and very exciting concept [48]. MIS, a concept unveiled to the managerial public for the first time only a year later, was already well on the way to becoming the single most widely discussed concept in the corporate computing world of the 1960s – promoted relentlessly by consultants, "systems men" (corporate staff specialists in administrative management), computer experts and computer manufacturers. Its advocates suggested that the best use of the computer, the only one to truly exploit its potential, was to build an enormous automated system capable of providing to each and every manager in an entire corporation every last piece of information necessary for the performance of their duties, in a timely fashion. It would reach, as Stone put it, "from board chairman to straw boss", and include sophisticated modeling and forecasting capabilities as well as simple factual reporting. [102, page 17].

Data processing was already well entrenched as the dominant name for administrative computing [47], but MIS enthusiasts suggested that this conservative and evolutionary approach wasted the power of the computer on mere clerical automation. MIS was intended to remove these expensive and unfamiliar machines from the too-pedantic hands of the accountant (who held "prejudices born of a lifetime of education and practice in the world of fine-ruled yellow analysis pads") and from former punched-card supervisor or "data processing technician", dismissed by Stone as a drone who would follow whatever instructions were placed in front of him.

These early, rather vague, concepts of data pools embedded the assumption that all relevant information, whether internal or external, past or future, economic or human, could be accommodated within a single structure. The 1950s had seen a sudden proliferation of discussion about information within a number of different fields. Shannon's mathematical theory of digital communication [98] was picked up as a powerful metaphor within the nascent meta-discipline of cybernetics. Librarians specializing in scientific and technical fields began to speak of themselves as information scientists [112], while researchers attempting to automate record searching started to call this work information retrieval [17, 76]. Glowing reports in *Fortune* magazine informed businessmen of the power of information theory [12] and of information retrieval [11]. In 1958, the combination of computers, operations research methods and simulation was first dubbed "information technology" [62]. Information was in the air, as a kind of universal solution to the various ills of business, science and government.

It was men such as Stone who first introduced managers to the idea of information as a generalized, abstract entity, separate from the forms, reports, files and memos in which it had previously been embodied. Stone recognized that a flexible and complete MIS could only be constructed if a firm's entire mass of paperwork could be computerized and integrated "to produce an interrelated body of useful data, or information." He suggested that "this body of data, a veritable 'bucket of facts,' [was] the source into which information seeking ladles of various sizes and shapes are thrust in different locations" [102, page 17]. Others, working with similar ideas, came up with other phrases over the next few years. Another consultant suggested that the office of the future would revolve around a "data hub", defined as "a central source of information that can serve as an instant inquiry station for executives who need data for decisions." [110] Representatives of Shell Oil spoke of the need for an "electronic data bank, or pool of information, from which reports of many types can be drawn." [51, 60].[3]

These buckets, pools and hubs seem quaint and rather unhelpful metaphors today, and indeed those trying to construct them using the technology of the 1960s were doomed to disappointment. Rather than flowing smoothly and easily into an ocean of knowledge, information instead coagulated messily around the small memories, tape drives, and inflexible file structures of early mainframes. Yet, if we can step back for a moment from the familiarity of the phase "data base", unknown in data processing circles as Stone spoke, is not a base of data even stranger, even more metaphorical, than a pool, bucket, hub or bank? These metaphors all serve to construct a particular version of information, in which the richness of social meaning that structures and supports information in its more specific manifestations (a parts list, a sales forecast, a letter of complaint) has been stripped away, leaving behind an inert substance that can

[3] It is worth pointing out in this context that Edgar F. Codd, creator of the relational data base model, informed the world of his invention in a paper entitled "A Relational Model for Large Shared *Databanks*" [33]. Even in 1970 the term was far from dead.

be stored, refined or piped as necessary. It implied that a single kind of technology or expertise, and therefore a single group of skilled professionals, could process information of any kind.

By the late 1960s, however, "data base" was a common expression in corporate computing circles, largely replacing the hubs, buckets and pools in which data had previously been rhetorically housed. This term was imported from the world of military command and control systems. It originated in or before 1960, probably as part of the famous SAGE anti-aircraft command and control network. SAGE [40] [56] was far more complex than any other computer project of the 1950s, and was the first major system to run in "real-time" – responding immediately to requests from its users and to reports from its sensors. As a result, SAGE had to present an up-to-date and consistent representation of the various bombers, fighters and bases to all its users. The System Development Corporation [10], a RAND Corporation group spun-off to develop the software for SAGE, had adopted the term "data base" to describe the shared collection of data on which all these views were based.

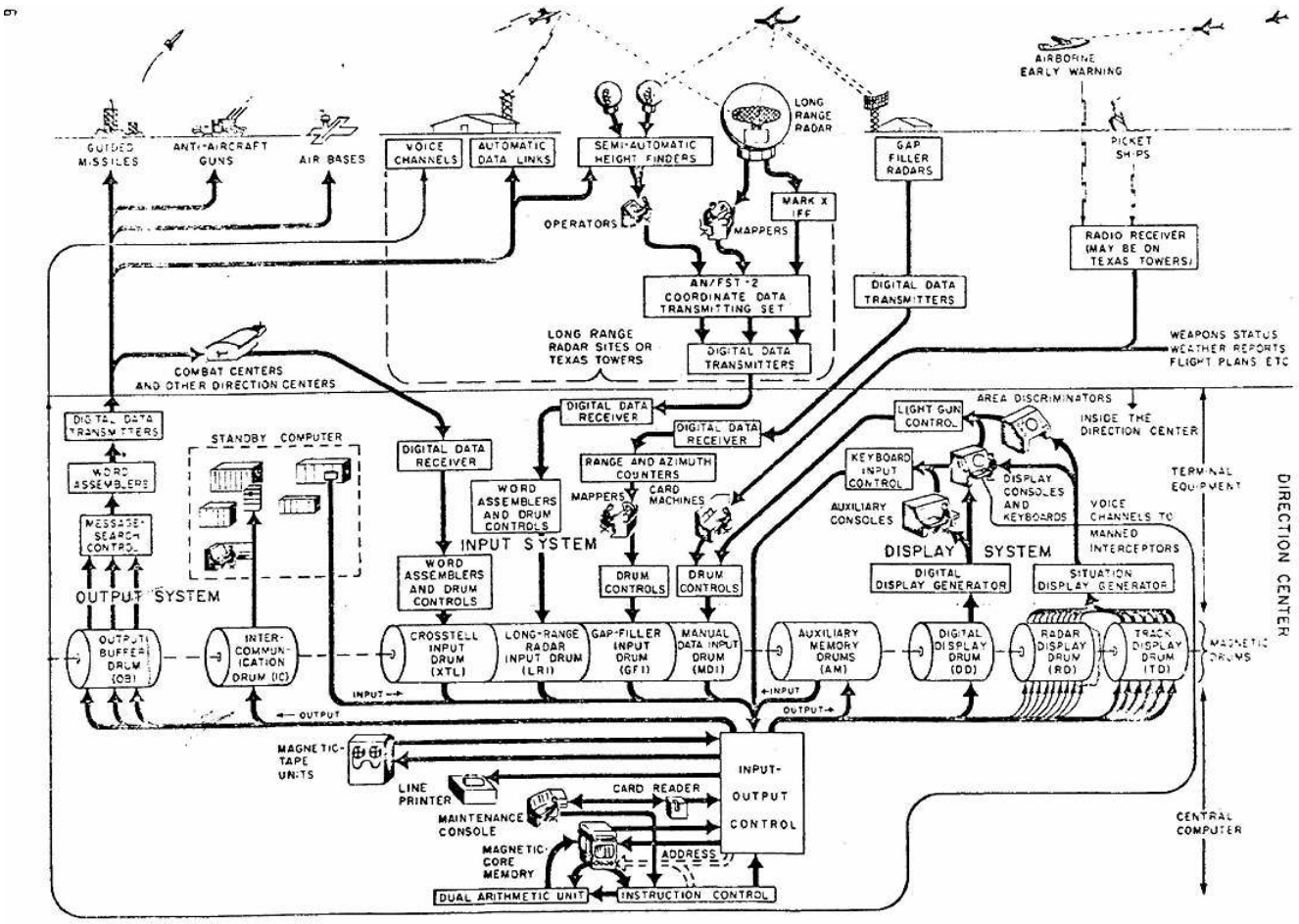


Figure 1: The SAGE system integrated data from many different sources (shown here as the "input system") and provided selective views of the overall situation via video consoles (shown here as the "display system"). Information was consolidated in the "central computer" (not shown). Developers of SAGE software may have been the first to use the term "data base" to describe a centralized body of data shared between many different subsystems. The image is taken from [58].

SDC actively promoted the data base concept for military and business use. Its interest in general purpose data base systems was part of its attempt to find new markets for its unique expertise in the creation of large, interactive systems. During the late 1950s and early 1960s, SDC held by far the world's largest concentration of programmers with experience in large-scale, real-time systems [94]. It paid particular attention to the fashionable area of "time-sharing" computer systems, in which one computer was used interactively by several people, each free to run whatever programs they required. Because computers were then large and expensive, time-sharing promised to make general-purpose, interactive computer use by non-specialists a commercial reality for the first time. SDC invested heavily in this area [4], and identified "computer-centered data base systems" as a key application of time-shared systems – hosting (in collaboration with military agencies) two symposia on the topic in 1964 and 1965. [104].

The SDC Data Base Symposia were crucial in spreading the data base concept beyond the world of real-time military contractors.

The approximately 185 participants at the second symposium included high-ranking military officials, business data processing celebrities, and corporate and academic researchers. Reporting on the event in *Datamation*, the leading trade magazine of business computing, Robert V. Head observed that data bases had already unleashed the "biggest single strike" of new jargon "since the great time-sharing goldrush of 1963," leaving potential users "sullen and down-trodden." He concluded by wondering whether it was "possible that users, led by the military, will surrender to these data base systems without a shot being fired in anger." [52, page 41]

It was around this time that the "data base" term made its first appearances in the ongoing discussion of management information systems. In 1965, Harvard accounting professor John Dearden was using the term "data base" to describe the truly important set of corporate facts and figures that had to be shared between different areas within a business [39]. Within the more technical literature it appeared as a means of pooling information from different files, so that each piece of data would be stored only once. Its great advantage would be "to permit categories of information to be added, deleted, expanded and otherwise revised, without completely redesigning the file or reprogramming the retrieval routines" [99, page 4].

The idea of the data base as a physical pool of data underlying an MIS was given an early, clear and highly influential statement by Head, who defined the data base as the bottom level of a pyramidal structure [53]. The data base pooled information from all the company's operational systems, and on top of it were erected reporting systems and models to inform higher level managers. [48, 45-50]. The metaphor fit very nicely with the idea of a data *base* supporting the rest of the information system. This obviated the need for systems experts to determine in advance exactly what information each manager would require. Instead managers could interrogate the data base and receive whatever information they needed. The data base was often called a "reservoir" of information [54, 61, 113, page 30].

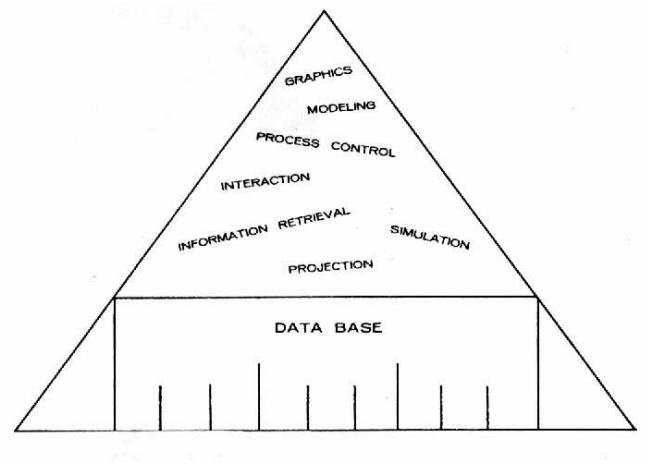


Figure 2: Head's concept of the data base as the support for other components of the management information system [53] was highly influential.

SDC's attempt to push the data base concept into civilian discourse worked well. The term data base carried some specific associations, based on the particular characteristics of firms like SDC and of military command and control projects. One of these associations was with the idea of real-time operation – the data base would be constantly and, if possible, automatically updated with current information gathered from a number of different sources. It was also assumed that, as in SAGE, a data base could be "interrogated" in real-time by its users, answering questions interactively within seconds. In addition, the data base would be shared among many different programs, each one using only a subset of the overall information contained within it.

In contrast, SDC's attempts to sell its own technology as a means of realizing this goal were not nearly as successful. SDC had used its data base symposia to showcase its own on-line systems [16], funded with military money, all of which ran on the special, and hugely expensive, computers developed for SAGE. [104]. SDC's most ambitious attempt to commercialize data base technology came with a system called CDMS (the Commercial Data Management System), a derivative of an earlier system called TDMS (Time-shared Data Management System) developed under contract from ARPA (Advanced Research Projects Agency) and given trial usage at military installations. These systems were intended to allow non-programmers to create data base structures, load data into them and then issue queries and retrieve their results on-line. Attempts to sell the TDMS computer program failed because it was expensive, needed a powerful computer all to itself, and could run only on SDC's own custom-developed operating system. Attempts to rent use of CDMS through terminals connected to centralized computers were equally unsuccessful. [10, pages 116-121, 101, 107].

In the late 1960s, the much discussed administrative data base remained a dream without any clear technological avenue of fulfillment. These early attempts to provide managers with interactive, on-line access to data stored in computer files suffered from a number of problems. These included the enormously expensive nature of the technology, a lack of interest on the part of most managers, and the largely unaddressed problems of taking data from all the routine, operational systems (payroll, accounting, inventory, billing and so on) and somehow integrating it and making it available inside the data base.

3. FILE MANAGEMENT SYSTEMS AND DATA PROCESSING

Besides the rather ill-defined concept of the "data base" the other main intellectual ingredient of the Data Base Management System, and the key technological foundation for the actual data base management systems of the 1970s, was the "file management system" (together with its close relation, the "report generator"). File management systems were intended to reduce the cost of producing routine administrative programs, and to make the finished programs easier to change and maintain. Report generation systems made it easier to produce printed reports based on particular criteria. These ideas, unlike the data base concept itself, were indigenous to the world of administrative data processing, where they had slowly evolved. Whereas the data base reflected a focus "blue sky" technology, on-line operation, scientific genius and enormous expense, these file management systems were initially oriented toward clerical tasks, were used and appreciated primarily by programmers and data processing supervisors, lacked features for interactive or on-line use, and did not cost much. Rather than glamorous managerial systems, they were humble but highly effective tools for computer technicians.

The need for such tools became apparent during the mid-1950s, as soon as computers were first applied to administrative tasks. Pioneering computer users had soon discovered that apparently simple clerical data processing activities, of the kind looked-down upon by enthusiasts for MIS, were far from trivial in practice. The pioneers of the late 1950s and early 1960s developed many new techniques and approaches as they struggled to contain programming and operations costs while maximizing flexibility. The techniques used to store data on tape were taken from existing punched card methods. Indeed the concepts of records, files, fields, special codes to mark the beginning and end of files, and the "merging" information from one file to another (all still ubiquitous in computer systems today) all have their origins in punched card systems.[4]

The first generation of American data processing installations spent much more and took far longer than expected to get their machines up and running. From General Electric's famous 1954 use of a Univac computer to automate payroll processing [81], data processing managers were shocked by the complexity of programming work and the rigid requirements computer technology imposed on areas such as data entry and the handling of special cases. Like punched card machines before them, early computers generally worked on one record at a time. The tiny internal memories of early computers, coupled with the inflexible, serial nature of tape storage, meant that a single major job such as payroll might require dozens of programs to be run one after another, each reading and writing information from several tapes [47].

File management systems evolved from the reuse of subroutines written to handle input and output tasks within application programs. Early computer programs included all the instructions necessary to specify the minute details of reading and writing information from tape or disk, and were forced to check regularly whether a particular record had yet been retrieved [69, 178-204]. Skilled programmers spent much of their time crafting routines to read records from tapes and print lines on paper, dealing each time with the many errors, synchronization problems, tape jams and so on that could frustrate their task. Programming groups soon hit on the idea of producing a single set of well written and reusable subroutines to handle these chores. Standard code was modified slightly to fit the particular situation and then inserted into each application program. Technological change also played a part. Application programs were closely tied to particular hardware configurations– even changing the tape drive used for temporary storage required considerable editing work, while adapting a program to make efficient use of more memory or additional tape drives involved a fundamental rewrite. The problem was compounded as companies attempted to reap the benefits of automation by using the output of one major application as the input to another, for example by linking their production scheduling system to their inventory control system, their accounts receivable system and their billing system. As computer manufacturers began to build more powerful capabilities into their data processing hardware, including buffers and auxiliary processing units to smooth the flow of data, the programming required to read and write records on tape became more complex. As a result, computer manufacturers began to supply their customers with standard functions to optimize these tasks [9, pages 181-185]. This made it easier to create new programs, but did little to help with other problems.

Another problem was the difficulty in extracting information from the computer – while daily, weekly or monthly runs of different parts of a payroll system might each produce voluminous printed reports, the only way to obtain a special report was to write another program. If a manager needed to tabulate data in a different way, or to include only a subset of the original records in the calculations, he could either wait for a programmer to become available or wade through the printout tallying records manually. By the late 1950s, the more innovative data processing teams had begun to address this through the creation of "report generation" programs, into which programmers fed a descriptions of the output desired and of the organization of the data inside the relevant "master files" and were rewarded with the desired reports. The work of General Electric's team at the Hanford Nuclear Reservation [72, 73] on its IBM 702 (IBM's first large computer designed primarily for administrative use) was particularly important in the establishment of these techniques.

File management systems had their origin in the use of similar techniques to create and update data files, as well as retrieve information from them. The most important initial areas were generalized routines to sort data into a particular order (a very important operation, and one that tape-based computers were very bad at doing compared to earlier punched card machines) and perform other routine maintenance operations on files. Because one major application might contain dozens of small programs, each reading and writing certain files, it might otherwise take Herculean efforts on the part of the programming staff to do something as simple as adding an extra digit to the employee number. By separating generalized file manipulation code from standardized descriptions of the record format used in each file, these approaches began to make it easier for programmers to modify record formats without completely rewriting programs. Such routines were written by the programming teams working inside computer using companies. In the early days of computing, it was common for system or utility programs of this kind to be shared freely, most notably through the SHARE user group established for users of large IBM computers [1]. During the late-

[4] By the 1940s, most punched cards included 80 columns of data, each one of which coded a single number or letter. Information within each card was grouped into fields, each occupying a fixed width within each record card. Consider a factory using punched cards to process its payroll [67]. Some fields needed only one column – for example sex (M or F). Other fields, such as last name, might be assigned a dozen columns. Each record would be punched onto one, or in some cases several, of the cards in the deck. The complete deck representing all the factory workers was known as a file, by analogy with conventional paper records. Each record card within the file had to follow exactly the same layout of fields, and to process a particular job the machine operators had to rewire the control panel of each machine (such as sorter, collator, or tabulator) to reflect this specific field layout. (Many jobs involved "merging" information from several files – for example combining wage information from the "master file" of personnel cards with the attendance information punched onto a weekly punched card by an IBM time clock). The concepts of file, record, and field were transferred directly to tape storage – though the records were now laid out sequentially along the strip of magnetic tape. Additional codes were introduced to mark the beginning and end of files and provide checks against corrupted data.

1950s SHARE coordinated efforts to develop General Electric's report generation system into more powerful systems for the IBM 709 called 9PAC, and a related project for the IBM 704 called SURGE.[5]

File management systems also proved an important niche for the nascent independent software package industry. Mark IV – the most successful product of the early independent software industry – was a file management system descended from report software produced for the Douglas Aircraft Company [43, 88, 89].

4. RANDOM ACCESS STORAGE

These file management techniques were very useful with tape storage, but when firms began to start storing their data on disk drives, the extra complexity of programming random access data storage and retrieval made their use almost essential. The disk drive was first offered as a standard option for most major computer systems in 1962 [2, 100, 109], though it had been available in a handful of IBM systems a little earlier. Whereas tape had previously been the only way of magnetically storing reasonably large files of information, it was suddenly possible to hold up to one billion characters of data on the disk drives connected to a single large IBM computer.

Disk technology progressed rapidly, and by the mid-1960s disks were standard options on many of the newly announced "third generation systems", along with operating systems, large memories, remote terminals, and other features marketed as the key to on-line application development [91]. Disk storage promised to bring the MIS dream of a fully integrated system much closer to reality. A large disk system appeared to be the physical bucket into which facts could be placed, and from which they could be checked, retrieved and updated by many different application programs.

In tape storage, records were generally sorted into a particular order and placed one after another along the tape. This was a fundamental limitation, because, as with today's video tapes, it might be necessary to wind through the entire tape to reach a desired spot. Users would still need to keep paper files, or leaf though big piles of routine printout, to get speedy access to a specific record. Disk drives, however, offered "random access" storage, giving almost instant access to any part of a disk. This promised to allow the speedy retrieval of specific data as needed, making it much easier to create special reports or to build on-line business systems such as the celebrated SABRE airline reservation system [34, 83]. Random access promised almost instant record retrieval, but although it was easy to order the computer to read a particular part of a disk (such as drive 4, platter 5, side 1, track 3, sector 15), there was no easy way to jump straight to a particular record (e.g. customer account 15274). One could, of course, keep the records sorted in order, but this would require an enormous amount of work rearranging the existing records every time a new one was added. Programmers experimented with a variety of strategies to arrange and index data on random access devices [69]. No single technique was suitable for all situations, and most of them were very complicated to program.

Another set of problems was caused by having several programs share a single disk, each using different program code to read and write records. Among these problems were the risk that an errant program might scramble an area of the disk holding information belonging to another, the overhead imposed by writing several different versions of the code required to handle complex indexing techniques, and the certainty that at some point the physical layout of the disk storage would be change (for example to shift a growing file to its own disk and expand the storage areas for the remaining ones) and all the programs would have to be modified at once.

The most obvious way to deal with this enormous increase in complexity was to rely on a new breed of generalized file management systems built to work with random access discs [22, 24]. These systems were intended to speed program development, reduce maintenance costs, shield application programs from the consequences of changes in the physical disk layout and make it easier to selectively retrieve records based on their contents.

By the end of the 1960s, every major computer manufacturer offered at least one piece of advanced file management software. These were usually based on the expansion of systems originally produced for use within a single organization. The most innovative, and influential, of these systems was General Electric's Integrated Data Store (IDS), created by Charles W. Bachman. IDS began life circa 1963, as part of an effort known internally as "Integrated Systems Project II." Its goal was the production of an integrated system for production control, flexible enough to be easily customizable by GE's many departments but powerful enough to give rapid results to queries on production scheduling and inventory levels while automatically placing orders and calculating the optimum order quantities. The resulting system, MIACS (sometimes, but not always, Manufacturing Information And Control System) relied on IDS to handle its data storage and retrieval needs. The project was very much in keeping with the early 1960s push to create integrated MIS systems, and Bachman recalls that top management were told that the project name stood for ***Management*** Information And Control System [6].

Manufacturing involves the assembling of multiple components into larger parts, which themselves usually serve as components in one of more kinds of larger assemblage. The need to solve this "parts explosion" problem made it particularly important for IDS to support the creation of linkages between different kinds of

[5] According to Charles W. Bachman, SHARE's SURGE project began as an attempt to modify the Hanford Report Generator for use with the IBM 704 but eventually took a different direction. The 9PAC system for the IBM 709, however, was produced by a different SHARE committee and appears to have been widely used. Although 9PAC was used exclusively with tape storage, it did permit the creation of hierarchical relationships between records. Child records were simply stored on tape immediate following their parent records. Data definitions for the file were stored in a header. [6] The project included a Report Generator and a Generalized File Maintenance system in its creators aimed to incorporate capabilities for calculated updates and modifications to the format of existing files. Specifications for the latter are in [70]. Their shared file structure is described in [71]. The series of SHARE Secretarial Distributions sent to all IBM 709 sites include much discussion of the project, including a series of drafts of documentation, requests for information, comments and suggestions. The 9PAC Subcommittee of the SHARE Data Processing Committee was formed in late May or early June of 1959.

record. While earlier systems had supported the idea of sub records, stored sequentially and hierarchically within master records, IDS was much more flexible. This generalized concept of linkages between record types, known later as the "network data model" was a major influence on early DBMS implementations.[6]

IDS was designed from the beginning for use with disk drives. It took over an entire GE 225 computer, providing basic operating system functions including an early implementation of paged virtual memory to squeeze out maximum performance from the 8K standard memory of the 225. The task scheduler for the MIACS application itself relied on IDS to store data. MIACS application programs (written in General Electric's own GECOM language) used simple instructions to navigate through the relationships between records and to STORE, GET, MODIFY or DELETE records one at a time. In the first implementation of IDS, a preprocessor replaced these special instructions with the appropriate strings of assembly instructions. However, efficiency concerns forced a switch to a different approach, where IDS performed this expansion interpretatively, combining the requested operation with metadata about the record type involved. This part of IDS remained resident in memory, waiting to deal with data requests from the application programs [6].

A few years later, around 1965, the first version of what eventually became IBM's Information Management System (IMS) was produced by IBM in collaboration with North American Rockwell to handle the proliferation of parts involved in the Apollo program [13]. The original version of this application, known as Generalized Update Access Method, ran on an IBM 7010 computer, and used a specialized hierarchical file management system to store its data on disk. IBM and NAA also developed a system called RATS (Remote Access Terminal System) so that interactive application programs could be accessed via terminals. In 1966 work began on a new version created to run as an application under OS/360 on the new System 360 machines, and it was this version that IBM distributed to other customers from 1968 onward. Like IDS, IMS was used by application programmers, using packaged procedures to embed data handling capabilities in their code. The OS/360 version allowed one memory resident copy of IMS to simultaneously service the data needs of multiple application tasks [26, 84].

General Electric offered an improved version of IDS to users of its computers, and IBM did the same with IMS. During the 1960s computer vendors "bundled" their software with hardware, using it as a free promotional tool to entice users into buying computers.

5. THE DATA BASE MANAGEMENT SYSTEM AND THE DBTG

The technological innovation represented by systems such as IDS was paralleled by conceptual developments. Until about 1968, the concepts of data bases and file management systems remained largely distinct. The data base was used interactively on-line, could be used by non-specialists and was closely associated with the MIS and the idea of a single huge reservoir of corporate information. File management systems were used primarily by programmers, to reduce development and maintenance costs for routine data processing applications. The most advanced file management systems were beginning to add features to make it easier to pool information from multiple files, and efforts were underway to add on-line access [18].

Combining the *data base* and the file *management system* created the Data Base Management System. The DBMS idea was shaped and promoted through the work of a body called the Data Base Task Group (DBTG), an ad-hoc committee of the computer industry group CODASYL (Committee On Data SYstems Languages). CODASYL's focus was the creation of data processing standards, and it is best known for its work designing and maintaining the COBOL programming language used for most business application programming from the late 1960s to the early 1990s. The DBTG was chaired by William Olle of RCA (then a manufacturer of mainframe computers) and its members were drawn from computer vendors, universities, consulting companies and a few large companies making heavy use of computers in their own business operations. Charles Bachman, the creator of IDS, was an early member of the committee and promoted the ideas he developed for IDS as the basis for its work.

As its name suggests, the DBMS was intended to be a new kind of product, extending the capabilities of existing file management systems to support the kind of advanced, on-line, interactive capabilities and huge integrated data stores associated with the data base concept. This was, in many ways, the endpoint of a natural evolution. The DBTG was dominated by the same manufacturers who were adding features to their file management systems and had begun to promote them as supporting, or even being, Management Information Systems [108]. The purpose of the DBTG was to define the capabilities of these new systems, and to develop new standards for them. Its creation was prompted by the realization within CODASYL that COBOL, while doing a great deal to standardize data storage on tape systems and to separate record definitions from program logic, was entirely inadequate when faced with the challenge of random access, disk based storage [80]. On its formation in October 1965 the DBTG had originally been called the List Processing Task Force (its name was changed only in 1967).[7]

In 1969 the DBTG released its first major report on what it now called "Data Base Management Systems". Despite lobbying by firms such as General Electric to get their own systems adopted as

[6] IDS is often, and with some justification, called the first data base management system. On a technical level, it was years ahead of its time and pioneered many important techniques. Powerful as it was, however, the initial version of IDS lacked some of the features associated with later systems and formalized in the CODASYL definition of a DBMS. Record definitions were punched directly onto cards in a special format rather than being defined and modified via a data definition language. It did not provide an interface for ad-hoc querying, or support for online access, since it was created purely to support the MIACS application. It did not provide different views or subsets of the overall database to different users. Neither did it support multiple databases simultaneously. Just as importantly, nobody at the time called it a data base management system. That concept itself did not exist at the time.

[7] The phrase "data base management system" was used at least once before the renaming of the DBTG, to describe IBM's forthcoming Generalized Information System (GIS) [18].

the basis for a new standard, the group decided that no single existing system came close to providing the range of features required. Instead, the group surveyed the strengths and weakness of existing systems, and began the attempt to standardize useful characteristics. Work continued, in part because the task group's parent committee was unsatisfied with the original results. In April 1971, a second and definitive major report [30] was issued and officially endorsed by CODASYL.[8]

The work of the DBTG provided both a broad conceptual outline for the data base management system, and detailed draft specifications for two specific parts of the over-all system (a data definition language for defining the data base structure, and a separate data manipulation language for accessing the data from within COBOL). It also outlined a way of giving individual programs access to selective or simplified versions of the full data base.

This conceptual framework for the DBMS ultimately proved more influential than the DBTG's detailed proposals. When the CODASYL work is mentioned at all today, it is usually for the propagation of the network data model. Since IDS, a commercial product, used this model prior to the DBTG's establishment this would seem a rather limited contribution to history. In fact, the DBTG appears to have created, or at the very least to have publicly defined for the first time, the very idea of the data base management system as we know it today.

The DBTG provided a new vocabulary with which to discuss these problems, including the separation of the "data definition language" used to define data base structures from the "data manipulation language" used by application programmers to work with the data itself.[9] Its final contribution was to insist that a standard DBMS allow more complex linkages to be established between different files (or, as they were now to be called, record types) within the same data base. The DBMS was intended to make these relationships (or, as the DBTG called them, "sets") as explicit and enforceable as previous file management systems had made the specification of fields within an individual file. Because most of the logic to maintain these relationships had previously been hidden within individual programs, placing relationships inside the DBMS along with the data itself ensured that all application programs and user requests would have access to them. The DBTG also decided that while the hierarchical approach used by systems such as IMS was good for some things, it proved unduly restrictive when applied to others. It instead specified a "network" model to represent these relationships, allowing the creation of more complex relationships between different groups of records.

Though most of the characteristics that the DBTG specified for a DBMS had already been demonstrated by at least one file management or data base system, it insisted that future systems must provide all of them. A DBMS was expected to provide the efficient, batch-based access for programmers and networked record-linking features that existing systems such as IDS specialized in. However, it was also expected to allow non-programmers to use a simple, specially tailored interface to query and update the data base directly – the province of systems such as Mark IV. Likewise, the DBMS was expected to support interactive on-line usage and batch operation with equal felicity.[31].

The term Data Base Management System, almost unknown before its adoption by the DBTG, spread rapidly from 1971 onward. It was applied retroactively to some existing systems, and used to describe virtually every new file management system, regardless of its fidelity to the specific ideas of the DBTG. This accompanies a great deal of publicity, as a flood of textbooks, technical articles and managerially-oriented pieces expounded on the potential of the data base. Following a traumatic transition to third generation equipment, many large corporations were now running powerful computers with large disk drives and flexible, multi-tasking operating systems and beginning to experiment with on-line terminals for data access. Meanwhile, the newly established market for independently produced packaged software was dominated by system software, particularly file management and data base management systems [49]. A 1973 article in *Infosystems*, the leading managerially oriented data processing publication, assured its readers that data base systems were like the aeronautical efforts of the Wright brothers: although carefully planned early efforts had "never developed much lift when applied to the practical realities of processing large files that had

[8] The specific proposals were controversial at the time, and several CODASYL members opposed them (including mainframe supplies IBM, RCA and Burroughs). [50] The IBM user groups SHARE and GUIDE went so far as to produce a rival report of its own. [23]. CODASYL's standards for the DBMS languages were not as successful as its work on COBOL, in the sense that no complete implementation of the specification was ever produced. The extensions to COBOL were reworked by a new standing committee into "Journal of Development" form as an official standard, published as [28]. Work on these standards continued into the 1980s, first through a new committee set up within CODASYL, and later at ANSI. This included a FORTRAN DML, to complement the COBOL DML in the earlier reports. [29] But these efforts to standardize a data definition language (DDL) failed to set a marketplace standard, in part because of the unwillingness IBM to commit to the network concepts inherent in the CODASYL model while its own flagship IMS product retained a hierarchical approach [85]. However, most of the advanced systems then under development were influenced to a more or less profound extent by ideas in the CODASYL reports – for a good summary of the most advanced commercial systems of the mid-1970s see [42, 44]

[9] The DBTG standardized terms such as "record" and "set" and "data base" and added some new ones, including "schema" (which remains ubiquitous today) to describe the logical format of data within the data base, and "sub-schema". A sub-schema (similar to what would be called a view in today's relational systems) allowed different users and applications to see only a portion of the overall database, allowing selective access to records and potentially shielding the application from changes in the underlying schema – a property referred to as "data independence." The DBTG also separated the Data Manipulation Language (DML) used to add, delete, update and retrieve particular records from the Data Definition Language (DDL) used to define the logical structure of the data base itself. While the DDL was to be a new and universally applicable language, the DML took the form of a set of additions seamlessly integrated into an existing programming language.

to be stored, indexed and sorted with live data" they were now poised to rise majestically into the air [93].

One immediate and dramatic result of the debut of the DBMS concept was a new surge of interest in the data base as the foundation of a company-wide management information system. During the late 1960s, a spate of bad publicity based on reports of delays and disasters among MIS pioneers had begun to raise doubts as to its practicality. The DBMS concept appeared as a technical savior for the idea of MIS as a single, all-encompassing system used directly by managers of all levels, and it featured prominently in many articles and textbooks of the early and mid-1970s. It functioned almost as a synonym for MIS. Richard Nolan, a Harvard Business School professor, consultant and one of the most prominent writers on computers and management during the 1970s, used a 1973 Harvard Business Review article to define the data base, rather boldly, as "a single pool or bank" where "all computer-readable data" is stored. He predicted that the long-awaited use of computers by senior executives was finally at hand, "from the union of the data-base concept and the corporation-model concept..." [78, pages 101 and 105]. As he observed the next year, "if the term Data Base or DB is used to replace the term MIS, the titles of recent articles are remarkably similar to the titles of MIS articles of several years ago" [79, page 27]. Many had simply seized on data base as a new and more palatable name for this "total" MIS.

Like the concepts of management information systems and of information retrieval, the idea of a data base was the intellectual product of a social movement trying to construct a new sense of information, as something that could be processed, retrieved and created using new bodies of scientific techniques. Like these other information concepts, the idea of a data base functioned in part to define a new area of professional authority. Considerable tension is apparent in the early 1970s, between those who, whether by virtue of temperament, practical experience or technical orientation, saw the DBMS as a practical tool to improve programmer efficiency and those who took this more utopian view of the data base and made few mentions of its technological underpinnings.

As one of the more practically oriented textbooks on the subject [66, page 22] explained, "A much-publicized but impractical idea of a data base says that a corporation keeps all its processable items of data in a large reservoir in which a diversity of data users can go fishing." The same idea had been given an early statement by Michael Scott Morton, founder of a prominent MIT group researching the managerial applications of computer technology, who in 1971 suggested that "the 'integrated' or 'company-wide' data base [was] a misleading notion, and even if it could be achieved would be exorbitantly expensive" [45].

These quibbles did not stop hopeful accounts of the data base as a technological marvel which would finally centralize and control information of all kinds, turning it from an abstraction into a solid organizational power base. This dream was enshrined in a new figure, the Data Base Administrator. According to one of the earliest descriptions, the DBA must "at once be technically qualified, if not inventive… he must encourage the users to work with him willingly and yet he will be forced to rule against their pet projects; he must represent both management and the users simultaneously; he must be all things to all people at all times." The author admitted that this role did "not exist as a formally established function in today's business" but considered its emergence imminent [64, page 12]. Nolan was still bolder: he believed [79,39] that the DBA would be responsible for "data as a resource… much broader than just computer-readable data," once the "data resource function [had been] carved out of the general management function." A consultant [63, page 9] wrote that the DBA should be "something of a superstar."

Discussion of the DBA makes the rift between managerially-oriented utopians and programmer-oriented pragmatists particularly apparent. Schubert, who at B.F. Goodrich had overseen a remarkably ambitious in-house DBMS development project, noted simply [95, page 47] that "Data base administration is accomplished by one or more technical experts who are knowledgeable in data base design and creation, operation of the data base management system, and the use of one or more data manipulation languages. The data base administrator must also be capable of working well with systems analysts, programmers, and computer operations personnel." It seems likely that this reflected practice in those firms actually using the technology rather than just talking about it; certainly, by the time DBMS technology became ubiquitous in the 1980s the DBA was a technical specialist rather than an information executive.

The idea of the "data dictionary" was given considerable discussion in the early 1970s. This was a central registry of the information gathered and produced by different parts of the business. By standardizing different representations of the same information, and establishing clear rules about who was responsible for each piece, companies could eliminate duplication and lay the groundwork for greater integration. This was originally seen as a managerial, rather than a technical, tool: one Arthur D. Little consultant noted that "in its simplest form, a data dictionary is a well-organized, up-to-date notebook containing basic information about data elements" [36, page 102]. But, as with the DBA, the data dictionary slipped from the managerial into the technical – after the term was applied to scores of software products in the late 1970s [21] it came simply to describe that portion of the DBMS where DML definitions were kept.

One IBM advocate of the data dictionary approach [20,23] likened data to money: "[o]nce management realizes the relationship of reliable data to corporate well-being, they will treat their data with the same care used to handle their cash." Nolan made a similar pitch in his book *Managing the Data Resource Function*, the title of which suggested that information, like people and money, was a vital resource of business and therefore deserved similar managerial attention [79]. Indeed, the claims made by Nolan that the DBA would be charged with overall responsibility for all corporate information, using computer technology where appropriate but ultimately claiming managerial rather than technical authority, directly prefigure those made more generally for the new position of Chief Information Office or CIO in the 1980s [103].

6. EARLY DBMS SYSTEMS IN USE

The DBMS enjoyed considerable practical success during the 1970s. By the end of the decade, most large computer installations had installed a DBMS package of some kind. Many of the most financially successful products of the independent software industry were DBMS or file management packages. Adoption of data base management software proved to be a boon to application programmers. In administrative applications of the kind traditionally carried out by corporate data processing departments,

an enormous amount of programmer time was taken up doing the things that DBMSs were supposed to automate. They made programs cheaper to develop, much easier to maintain, and facilitated the integration of different business tasks. Data base management technology as defined by the DBTG was very good at dealing with very uniformly structured, hierarchical data of the kind found on administrative forms.[10]

Yet the DBMS never quite lived up to the expectations of people like Nolan, who saw it as a managerial panacea. Indeed, the managerial hype that developed around DBMS technology may have made it hard for firms to make informed technical decisions. As early as 1973, a report [35] by two Booz, Allen & Hamilton consultants suggested that both software and the hardware needed remained immature, that little experience so far existed in its use and that the generalized features offered by the DBMS brought a hefty performance penalty and might well trigger the purchase of more memory or a new processor unit. Most of the true costs were hidden, particularly the staff requirements. As they put it, "Some DBMSs are as complex as the operating system which services them. Also, this group must continuously apply and test new program fixes and new features to keep the system 'alive and well.' It is not uncommon to see a small systems programming team double or even triple as the result of a DBMS" [35, page 74]. Later reports, in [97], suggest that these problems continued for several years, and that many firms installed DBMS packages because of a "bandwagon" rather than a careful and informed evaluation. Despite direct access by executives being a theoretical keystone of the data base as an MIS tool, no surveys of the early 1970s were able to find any firms where the data base was used directly by managers, or even by analysts [78, page 113].

Companies keen to get their hands on a DBMS had to go to considerable lengths. Richard F Schubert of chemical firm B.F. Goodrich had been part of the DBTG, and led his company into implementing its own system IDMS based on a stripped down version of the CODASYL proposals. It was used to support batch mode applications such as billing and accounting as well as on-line access to order entry and its inventory of finished goods [57]. In 1973, Goodrich sold the rights to IDMS to marketing savvy entrepreneur John Cullinane, who by the early 1980s had built an eponymous software firm, one of the era's largest and fastest growing, around it [68, 242-246]. Few companies were prepared to go this far to get a DBMS, and indeed experts of the early 1970s agreed that the exceptionally complex and generalized nature of the technologies involved made the selection of a good package far more sensible than trying to develop a system in-house.

Even among firms acquitting the most advanced DBMS packages, on-line use was limited and managerial applications rare. Let us consider two examples of firms using commercially supplied DBMSs in the early 1970s. McDonnell Douglas, using IBM's IMS system, claimed to have created a centralized data base containing all the information previously stored in 264 files covering things like spare parts, production scheduling, bill of materials handling, and inventory management. This made it much easier to change the 95 existing programs that relied on these files, and to set up automatic cross references between different records and, it hoped, to move toward on-line operation in the future [55]. The second firm, much smaller, was devoted to accounts receivable processing for doctors. It used an DBTG influenced DBMS on its Xerox computer to lower its daily processing times shorter for updates and design its new program more rapidly. The results pleased it, despite the fact that the DBMS consumed a large part of the computer's memory and used ten times more processor capacity than the tape based version. It had moved cautiously into on-line operation – while records were retrieved using terminals, all updates were queued and applied at night while the system was off-line in the belief that this "greatly reduces the possibility of a catastrophic loss of data" [14, page 63].

According to a 1975 survey of large industrial firms [90], about one third were using some kind of advanced file or data base management system. Of that third, around half were using systems intended for direct ad-hoc querying by non-programmers, such as MARK/IV, and half were using systems designed to integrate with the conventional programming languages such as COBOL. Hybrid systems, of the type envisioned by CODASYL, had yet to make much impact. Only about a quarter of the systems were used primarily for on-line access, and only two firms claimed to have implemented a data base for the entire firm, though most reported using it for multiple areas of the business. This was very slow to change. Five years later a survey of management information systems in thirty two large corporations found that most of these companies had now installed powerful DBMS packages [27]. Yet when the researchers looked at the actual use made of these systems they found that, "The users surveyed were only beginning to develop DBMS applications.... This is possibly because of the difficulties involved in developing and controlling such activities" [27, page 28].

Even products designed explicitly for use by non-specialists found their main markets to be among data processing specialists. Because they cost less and could run on more modest hardware, file management systems remained more widely used than fully fledged DBMSs. The 1975 survey by Powers found that 41% of firms using these packages reported that information could only be retrieved with the aid of a programmer. Unlike the more powerful systems designed primarily for application programmers to use, these systems were still used primarily (in 77% of firms) with files stored on tape rather than on disk. These systems still worked with individual files rather than vast integrated data bases – indeed, 55% of their users had not even begun to integrate files to remove redundant information.

During the 1970s, the MARK IV file management system became the most successful single product in the admittedly short history of the industry: the first to reach the milestones of $1 million, $10 million, and $100 million in cumulative sales. Compared to the DBTG proposals, its capabilities were modest. Its initial appeal was straightforward: first it was highly efficient in batch operation, and second it had been designed for use by non-

[10] Like file management systems before them the new systems still demanded that each record in a file (or "record type") include exactly the same fields, each populated with data in exactly the same format. In addition, because relationships (or "sets") were specified in the Data Definition Language, and so built into the database, the data base designer was forced to specify a complete and coherent set of linkages between different files – something that proved essentially impossible to do for the kind of large-scale, complete and multifunctional data bases envisioned by MIS proponents.

programmers. Requests for data were entered onto one of four simple paper forms, and then keypunched into computer form for later processing. But even MARK IV found its main audience among programmers. As time went by, development of MARK IV focused more and more on the needs of full time programmers, who used it as a foundation for the construction of complex application programs. An official company history credited this process to the influx of data processing specialists into its "IV League" user group, which ensured that their opinions "overwhelmed the voices of the non-programming end users" in the company's planning [43, page 9.26]. The proceedings of this group suggest that non-specialists found advanced work harder than had been expected. According to an Eastern Airline representative, while most of its 200 users were "a complete new breed of coders… non-programmers, little or no data processing background," attempts to train them in information retrieval techniques without giving an understanding of what went on in "the mysterious black box" of the computer had failed. Contrary to their expectations, "The only users able to move into extended capabilities with any degree of success were those with some data processing background" [65, Appendix F].

While there was a substantial demand for products that would let non-specialists produce computerized reports without the assistance of programmers, the leading DBMS systems did not do a good job of meeting this. One of the most successful software products of the 1970s, Pansophic's Easytrieve, was an easy to use report generation system designed to extract information from files and data bases. Easytrieve thrived in competition with more complex DBMS and file management software, and many firms purchased the optional modules needed to use it in conjunction with the most powerful DBMSs [87].

By the end of the 1970s, then, it was clear that DBMS technology had failed to live up to the hopes vested in it by its more managerially focused promoters. While powerful DBMS systems were now common in large corporations, few were being used to support new kinds of managerial application. Even the most sophisticated DBMS systems were used mostly in batch mode rather than on-line, and by programmers rather than managers. The data base management system was more of an improved file management system. Massive, integrated data stores remained very hard to construct, while interactive computer models of the kind anticipated by advocates of MIS remained conspicuous by their absence.

7. THE DATA BASE MANAGEMENT SYSTEM SINCE 1980

In 1973, Charles W Bachman was awarded the Association for Computing Machinery's Turing Medal – the most prestigious award in computer science. The citation singled out his creation of the pioneering IDS system (which it retroactively termed a DBMS) and his work on the DBTG to incorporate these ideas into its specifications. This award was in itself an important event, representing a new level of acceptance among computer science researchers of data base problems as intellectually respectable subjects of inquiry alongside better established areas such as numerical analysis, compiler theory and the theory of algorithms. The event is better remembered, however, for Bachman's speech [7]. Entitled "The Programmer As Navigator," it developed the idea that the shift to DBMS technology represented something akin to the Copernican revolution – in that the work of programmers would now revolve around the data base rather than the hardware of the computer. Though this prophecy took several decades to come true, knowledge of data base systems has now become a fundamental requirement for virtually all administrative applications programming, systems analysis and advanced web design work. But, as its title also implied, the impact of generalized DBMS would be much greater for programmers than for managers.

The acceptance of the DBTG concept of a data base management system thus implied a new and more concrete vision of what a data base was – basically a body of electronic data that could be managed by a data base management system. As such, the commercial success of DBMS packages supported the growing prestige of corporate computing staff, against attempts by information scientists and documentationalists [5] to turn the library, rather than the computer room, into the heart of any corporate information system. Despite the MIS influenced hopes of the 1970s that a DBMS could be the heart of a system including all corporate information, it proved adept at handling only a small subset of this material. The data base, as realized through an extension of existing file processing tools, embodied the highly structured, administrative transaction-oriented view of information held by data processing staff and computer vendors.

The narrowing of the data base concept, and its close association with the DBMS, also represented a shift away from the idea, implicit in much earlier discussion of information retrieval, that all important information was scientific or at least was amenable to the same retrieval techniques as scientific information. The data base concepts pioneered by elaborate, military systems of the 1960s such as SDC's TDMS – on-line access, flexibly structured data, interactive definition of data formats by users, played little part in the leading commercial systems of the 1970s. Neither was there a significant commercial market for products based on these technologies. Though the industrial research budgets of leading corporations might have paid for subscriptions to the newly available on-line scientific data bases of the 1970s [46], the managers and computing departments of the same companies had little interest in using these technologies to manage their own information. The commercial emergence of on-line information services, discussed in great detail in [16], appears to have taken place in almost complete isolation from work on DBMS packages.

The DBMS concept proved far more important and longer lasting than the particular methods for its realization put forward by the CODASYL DBTG. During the 1970s a new approach, the relational model [25, 33], gradually gained acceptance among database researchers. The relational model was far more conceptually elegant and flexible than the network model endorsed by CODASYL, which proved both restrictive (because relationships must be specified when the data base is designed) and insufficiently abstracted from the physical storage of data (programmers were still forced to write code to navigate explicitly from one record to another when working with linked data). Because the relational model shifted the responsibility of specifying relationships between tables from the person designing them to the person querying them, it permitted tables to be joined in different ways for different purposes. This turned out to be necessary (if not sufficient) for the establishment of large, general purposes data bases shared between different departments and computer systems. The relational model has also been praised for its non-procedural nature – further separating the user from the

physical storage mechanisms involved [75]. This simplified programming and insulated application code from changes in the database structure. Accepting his own Turing award in 1981 for his development of the relational data base model, Edgar F. Codd suggested that the CODASYL network model had forced the programmer to become too much of a navigator, at too low a logical level [32].

Use of early DBMS systems was highly concentrated. According to internal reports prepared by one software firm, as late as 1981 TOTAL, the market leader, had just 4,171 installations while IBM's IMS won second place with an estimated 1,500 [82]. The first widely used relational DBMS, Oracle, was launched in 1980 and found an early niche in the rapidly growing market for minicomputer systems. During the 1990s, relational systems gained the power and maturity to gradually edge out earlier mainframe products such as IMS, though even today the transition is far from complete. At the same time, the increasing power of personal computer systems opened new niches for DBMS technology on desktop computers and inexpensive departmental servers. Almost every custom business application produced during the past decade relies on a relational DBMS to store and retrieve data. Relational DBMS systems are widely used on personal computers. Indeed, Microsoft now bundles a version of its powerful SQL Server DBMS with the "professional" editions of its Office suite, and has even adapted it for use with its Pocket PC hand held computers. Microsoft has long aimed, though so far without success, to replace the conventional file system and the email repositories found on today's Windows operating systems with a multitalented DBMS. In some ways, the DBMS has indeed become a universal container for computer data.

8. CONCLUSIONS

The data base management system provides an interesting example of the tensions hidden behind phrases such as "information technology." The progression of the concepts of data base and data base management system over the 1960s and 1970s demonstrate an unmistakable tension between the rather limited and technically focused achievements of actual information systems and the universal, almost utopian claims that information problems can be defined, and therefore solved, for the general case if only the right tools or technologies can be deployed. The technologies of the file management system, however much improved, could never realize the grand dreams set forward for corporate data bases as universal sources of information. While the invention of the DBMS concept initially revived hopes for the creation of all-powerful data bases, in the longer term its effect was to redefine the very concept of a data base (or as we now say, database) as the contents of a DBMS.

Despite their remarkable ubiquity, DBMSs based on the relational model continued to incorporate the same assumptions about information as earlier file management systems. In particular, the complexity of relational query construction meant that to query and update the data base still required the involvement of a programmer, a specially written application program, or trained specialist. The designers of the now-standard SQL language had assumed that replacing algebraic characters with words such as "SELECT" would make it easy for managers to write their own queries [74], but the complexity and rigor could not be removed so easily. And although the relational model made it easier to join tables together in different ways, data base designers still had to specify the exact format of each column within the table, and include exactly the same fields in each row.

As a result, the DBMS was very well suited to the bureaucratic records for things such as payroll administration, because each record included the same pieces of data (years of service, SSN, hourly rate, overtime status and so on). It made it very simple and efficient to update information, and so is well suited to administrative systems where records are constantly updated. On the other hand, it was entirely useless for representing and searching less rigidly formatted data, such as full-text records, correspondence, or even scientific abstracts. There has certainly been no shortage of interest by database researchers in the design of alternative and more flexible models. Many of these have been promoted in commercial products, including object oriented database systems [114], multi-valued databases and other approaches. Indeed, "post relational" has become a marketing buzzword during the last decade, and just like "relational" in the 1980s, and indeed "data base management system" in the 1970s, it has been applied to such a broad range of products as to thoroughly blur its actual meaning [37].

The point remains, however, that today's dominant data base technology still includes rigid concepts of fields and data types inherited from punched card systems and remains far from living up to the vision of a universal repository for data of all kinds. Only with the rise of the World Wide Web in the mid-1990s did widespread attention turn back to the indexing and management of huge amounts of natural language information. Systems such as AltaVista and, more recently, Google have proved remarkably adept at returning relevant results from a sea of unstructured data. However, these technologies remain quite distinct from mainstream DBMS systems.

As DBMS use proliferated, firms found themselves unable to integrate all corporate data into a single pool in the manner promised by early data base advocates. When DBMS technology achieved almost universal use, large firms were left with hundreds or thousands of disconnected and duplicated data bases and no easy way to merge them. Data warehousing, one of the leading obsessions of corporate IT departments and consulting firms of the mid- and late-1990s, was an attempt to construct enormous read-only data bases for reporting purposes in which all data was linked and reformatted into a standard form. Firms intended to use these buckets of facts for "data mining" and the provision of "business intelligence" to best their competitors [38]. The corporate data pools imagined forty years ago have inched ever closer to reality, yet the messy reality of specialized, limited and inflexible data storage technologies continues to contrast with the pristine simplicity of the original vision.

9. APPENDIX: FILE AND DATA BASE MANAGEMENT PACKAGES OF THE 1960S AND EARLY 1970S

System	Produced By	Approximate release date	Notes
9PAC	SHARE	1959	Consisted of generalized File Maintenance and Report Generation systems. Developed cooperatively by a group of IBM 709 installations through the SHARE organization, and heavily influenced by the earlier generalized report generation system produced by General Electric's Hanford site. 9PAC used tape storage, but included record descriptions in the file header and allowed record hierarchies within a file.
GIRLS (Generalized Information Retrieval and Listing), MARK I-III	Douglas Aircraft	1962-196?	A series of programs developed by John A. Postley, beginning with generalized reporting and evolving into a batch based file management system intended for use by non-programmers. [43, 88, 89]. Commercialized as Mark IV (see below).
IDS (Integrated Data Store)	General Electric (later Honeywell Information Systems)	1963 onward for internal use. GE product from mid-1960s.	File management system, evolved from internal GE application used to track inventory levels. Batch based, used by programmers, often with COBOL. Pioneered the creation of links between records in different files. Through its creator, Charles W Bachman[11], was a major influence on CODASYL specifications. [8, 22, 92]. With Honeywell takeover of GE computer business, ran on Honeywell H600 and H6000 series machines.
TDMS/CDMS (Time-shared/ Commercial Data Management System)	SDC	1967, 1969	On-line timeshared system, allows non-programmers to create data base definitions, load data, and issue queries. Ran on IBM/360 series. TDMS by military. See [10, 116-121] and [107]. Renamed CDMS and offered as a commercial product for the IBM/360 series from 1969, but required SDC's own operating system. Then as a flagship service for SDC's nationwide network of SDC Data Centers. [10, 16, 19, 101].
IMS (Information Management System)	IBM	1968 as product	Evolved from application produced in collaboration with Rockwell for use at NASA to tack components for the Apollo program. [13]. Used hierarchical data model, versus the network model of IDS, but supported online applications. Still in use on some mainframes.
MARK IV	Informatics General	1968	File management system, developed from above. Intended for batch mode use by non-programmers. Simple interface for creation, updating and querying of files. Highly successful product, though use came primarily from programmers. On-line support late and limited. [19, 101, 111]
GIS (Generalized Information System)	IBM	1969	Intended to function in batch or on-line mode. Oriented toward ad-hoc querying by non-specialists. Ambitious but repeatedly delayed. [18] [19] [61].[86]
IDMS (Integrated Database Management System)	B F Goodrich initial work. Cincom as product.	Initial development 1969-1972.	Developed internally by B F Goodrich for internal use on IBM/370 mainframe. Powerful DBMS heavily influenced by CODASYL concepts, including the network data model, and by previous experience with IDS. Worked primarily in batch mode, with support for on-line applications. Marketed by Cullinane Corporation from 1973 onward. One of the most successful DBMS products of the late 1970s and early 1980s. [57, 59, 68, 95, 96].
TOTAL	Cincom	1970	Firm founded by former IBM salesman. TOTAL was an early CODASYL influenced DBMS, based on the network data model. [3].

[11] The Charles W. Bachman Papers (CBI 125), Charles Babbage Institute, University of Minnesota, Minneapolis include considerable material on IDMS, though most of this dates from after Bachman joined Cullinane in 1980.

ADABAS	Software AG	1970	Highly efficient DBMS using "inverted file" structures to index files, providing high speed batch mode capabilities for application developers. ADASCRIPT query language intended for use by non-specialists. Still sold.
DMS 1100	Univac	1971	DBMS based on CODASYL concepts, produced by Univac for users of its 1100 series machines. [15, 105]
EASYTREIVE	Pansophic (distributed and eventually acquired)	1973	An easy to use querying and report generation system, combined with a tape based file management system. Highly successful, and one of the top ten products of the independent packaged software industry during the 1970s. Its success illustrates the continuing demand for simple systems, and the popularity of optional modules to connect it to IMS and IDMS as an alternative front-end system for querying shows the limitations of early DBMS systems in handling ad-hoc queries and reports. [87]
System 2000	MRI	Early 1970s	Company founded by University of Texas researchers. Hierarchical system, but influenced by CODASYL model. Offered for IBM, CDC, Univac machines. Included report capabilities, good support for ad-hoc queries. [106]

10. ACKNOWLEDGMENTS

Thanks to Rick Snodgrass for introducing this paper to SIGMOD Record, to Mary Ellen Bowden for encouraging me to write it in the first place, to Boyd Rayward for his close attention and helpful comments through multiple drafts, to ACM SIGMOD for funding my oral history interview with Charles W. Bachman and to the Computer History Museum for supporting my oral history interview with Robert L. Patrick.

11. REFERENCES

[1] Akera, A. Voluntarism and the Fruits of Collaboration. *Technology and Culture*, *42*, 4 (October 2001), 710-736.

[2] Anonymous (ed.), *Disc File Applications: Reports Presented at the Nation's First Disc File Symposium*. American Data Processing, Inc., Detroit, 1964.

[3] Anonymous MIS, the Impossible Dream? *Infosystems*, *20*, 2 (February 1973), 70.

[4] Anonymous A panel Discussion on Time-Sharing. *Datamation*, *10*, 11 (November 1964), 38-44.

[5] Aspray, W. Command and Control, Documentation, and Library Science: The Origins of Information Science at the University of Pittsburgh. *IEEE Annals of the History of Computing*, *21*, 4 (October-December 1999), 4-20.

[6] Bachman, C.W. *Oral History Interview by Thomas Haigh, 25-26 September 2004, Tucson, AZ, 2004*, Personal collection of author, to be published in ACM Digital Library (details unknown).

[7] Bachman, C.W. The Programmer as Navigator. *Communications of the ACM*, *16*, 11 (November 1973), 653-658.

[8] Bachman, C.W. and Williams, S.B. *The Integrated Data Store, A General Purpose Programming System for Random Access Memories*, May 1 1964, Charles W. Bachman Papers (CBI 125), Charles Babbage Institute, University of Minnesota, Minneapolis.

[9] Bashe, C.J., Johnson, L.R., Palmer, J.H. and Pugh, E.W. *IBM's Early Computers*. MIT Press, Cambridge, MA, 1986.

[10] Baum, C. *The System Builders: The Story of SDC*. System Development Corporation, Santa Monica, 1981.

[11] Bello, F. How to Cope with Information. *Fortune*, *62*, 3 (September 1960), 162-167, 180-182, 187-189, 192.

[12] Bello, F. The Information Theory *Fortune*, 1953, 136-141, 149-150, 152, 154,156, 158.

[13] Blackman, K.R. IMS Celebrates Thirty Years as an IBM Product. *IBM Technical Journal*, *37*, 41998), 596-603.

[14] Blanchard, J.S. We Bet Our Company on Data Base Management. *Datamation*, *20*, 9 (September 1974), 61-65.

[15] Borgerson, B.R., Hanson, M.L. and Hartley, P.A. The Evolution of the Sperry Univac 1100 Series: A History, Analysis, and Projection. *Communications of the ACM*, *21*, 1 (January 1978), 25-43.

[16] Bourne, C.P. and Hahn, T.B. *A History of Online Information Services: 1963-1976*. MIT Press, Cambridge, MA, 2003.

[17] Bowles, M.D. The Information Wars: The Two Cultures and the Conflict in Information Retrieval, 1945-1999. In Bowden, M.E., Hahn, T.B. and Williams, R.V. eds. *Proceedings of the 1998 Conference on the History and Heritage of Scientific Information Systems*. Information Today, Inc., Medford, NJ, 1999, 156-166.

[18] Bryant, J.H. and Semple, P. GIS and File Management. In Association for Computing Machinery ed. *Proceedings of the 21st National Conference*. ACM, New York, 1966, 97-107.

[19] Byrnes, C.J. and Steig, D.B. File Management Systems: A Current Summary. *Datamation*, *15*, 11 (November 1969), 138-142.

[20] Cahill, J.J. A Dictionary/Directory Method for Building a Common MIS Data Base. *Journal of Systems Management*, *21*, 11 (November 1970), 23-29.

[21] Canning, R.G. The Data Dictionary/Directory Function. *EDP Analyzer*, *12*, 10 (November 1974).

[22] Canning, R.G. Data Management: File Organization. *EDP Analyzer*, *5*, 12 (December 1967).

[23] Canning, R.G. The Debate on Data Base Management. *EDP Analyzer*, *10*, 3 (March 1972), 1-16.

[24] Canning, R.G. New Views on Mass Storage. *EDP Analyzer*, *4*, 2 (February 1966).

[25] Chamberlin, D.M. Relational Data-Base Management Systems. *ACM Computing Surveys*, *8*, 1 (March 1976), 43-66.

[26] Charles Babbage Institute *Software History Dictionary Project: Information Management System (IMS)*, Charles Babbage Institute, February 17 2004 [cited April 15 2006] http://www.cbi.umn.edu/shp/entries/ims.html

[27] Cheney, P.H. and Lyons, N.R. MIS Update. *Data Management*, *18*, 10 (October 1980), 26-32.

[28] CODASYL Data Description Language Committee *CODASYL Data Description Language: Journal of Development, June 1973*. U.S. Govt. Print. Off., Washington, DC, 1974.

[29] CODASYL FORTRAN Data Base Manipulation Language Committee CODASYL FORTRAN Data Base Facility *Journal of Development*, 1977.

[30] CODASYL Systems Committee *CODASYL Data Base Task Group: April 1971 Report*. Association for Computing Machinery, New York, 1971.

[31] CODASYL Systems Committee *Feature Analysis of Generalized Data Base Management Systems*. [n.p., Distributed by the Association for Computing Machinery], New York, 1971.

[32] Codd, E.F. Relational Database: A Practical Foundation for Productivity. *Communications of the ACM*, *25*, 2 (February 1981), 109-107.

[33] Codd, E.F. A Relational Model for Large Shared Databanks. *Communications of the ACM*, *13*, 6 (June 1970), 277-390.

[34] Copeland, D.G., Mason, R.O. and McKenney, J.L. SABRE: The Development of Information-Based Competence and Execution of Information-Based Competition. *IEEE Annals of the History of Computing*, *17*, 3 (Fall 1995), 30-57.

[35] Cuozzo, D.E. and Kurtz, J.F. Building a Base for Data Base: A Management Perspective. *Datamation*, *19*, 10 (October 1973), 71-76.

[36] Curtice, R.M. Some Tools for Data Base Development. *Datamation*, *20*, 7 (July 1974), 102-106.

[37] Date, C.J. What do you mean, 'Post Relational'? 2000 [cited April 20 2004] http://www.pgro.uk7.net/cjd1a.htm

[38] Davenport, T.H. Competing on Analytics. *Harvard Business Review*, *84*, 1 (January 2006), 98-107.

[39] Dearden, J. How to Organize Information Systems. *Harvard Business Review*, *43*, 2 (March-April 1965), 65-73.

[40] Edwards, P. *The Closed World: Computers and the Politics of Discourse in Cold War America*. MIT Press, Cambridge, MA, 1996.

[41] Elmasri, R. and Navathe, S.B. *Fundamentals of Database Management Systems*. Benjamin/Cummings, New York, 1989.

[42] Flynn, R.L. A Brief History of Data Base Management. *Datamation*, *20*, 8 (August 1974), 71-77.

[43] Forman, R.L. *Fulfilling the Computer's Promise: The History of Informatics, 1962--1982*. Informatics General Corporation, Woodland-Hills, CA, 1984.

[44] Fry, J.P. and Sibley, E.H. Evolution of Data-Base Management Systems. *ACM Computing Surveys*, *8*, 1 (March 1976), 7-42.

[45] Gorry, G.A. and Morton, M.S.S. A Framework for Management Information Systems. In Nolan, R.L. ed. *Managing the Data Resource Function*. West Publishing Company, St. Paul, 1974, 87-105.

[46] Hahn, T.B. Pioneers of the Online Age. *Information Processing and Management*, *32*, 11996), 33-48.

[47] Haigh, T. The Chromium-Plated Tabulator: Institutionalizing an Electronic Revolution, 1954-1958. *IEEE Annals of the History of Computing*, *23*, 4 (October-December 2001), 75-104.

[48] Haigh, T. Inventing Information Systems: The Systems Men and the Computer, 1950-1968. *Business History Review*, *75*, 1 (Spring 2001), 15-61.

[49] Haigh, T. Software in the 1960s as Concept, Service, and Product. *IEEE Annals of the History of Computing*, *24*, 1 (January-March 2002), 5-13.

[50] Hare, V.C., Jr A Special Report on the SIGBDP Forum: 'The New Data Base Task Group Report". *Data Base*, *3*, 3 (Special Issue 1971), 1.

[51] Haslett, J.W. Total Systems - A Concept of Procedural Relationships in Information Processing. In Meacham, A.D. and Thompson, V.B. eds. *Total Systems*. American Data Processing, Inc., Detroit, MI, 1962, 16-19.

[52] Head, R.V. Data Base Symposium. *Datamation*, *11*, 11 (November 1965), 41.

[53] Head, R.V. Management Information Systems: A Critical Appraisal. *Datamation*, *13*, 5 (May 1967), 22-27.

[54] Head, R.V. MIS-II: Structuring the Data Base. *Journal of Systems Management*, *21*, 9 (September 1970), 37-38.

[55] Hollenbach, R. An Application of a Data Base System. *Data Management*, *11*, 9 (September 1973), 30-31.

[56] Hughes, T.P. *Rescuing Prometheus*. Pantheon Books, New York, 1998.

[57] Huhn, G.E. The Data Base in a Critical On-Line Business Environment. *Datamation*, *20*, 8 (September 1974), 52-56.

[58] International Business Machines Introduction to the AN/FSQ-7 Combat Direction Central and AN/FSQ-8 Combat Control Central, Ed Thalen, 1965 [cited April 22 2006] http://ed-thelen.org/SageIntro.html

[59] Karasz, P. and Hoelscher, C. History of IDMS, Australian IDMS Users Group, 1999 [cited July 5 2002] http://users.senet.com.au/~cherlet/idmshist.html

[60] Keller, A.E. The Man Behind Systems at Shell Oil. *Business Automation*, *7*, 2 (February 1962), 20-24.

[61] Kircher, P. Breakthrough in Management Information Systems. *Journal of Data Management*, *7*, 2 (February 1969), 28-31.

[62] Leavitt, H.J. and Whisler, T.L. Management in the 1980s. *Harvard Business Review*, *36*, 6 (November-December 1958), 41-48.

[63] Luke, J.W. Data Base Systems: Putting Management Back In The Picture. *CSC Report*, *9*, 11975), 8-12.

[64] Lyon, J.K. The Role of the Data Base Administrator. *Data Base*, *3*, 4 (Winter 1971), 11-12.

[65] Mark IV User Group *Proceedings of the MARC IV User Group: meeting IX*, Jan. 25-27, 1971, San Francisco, CA, 1971, Evan Linick Collection of Proceedings of the Mark IV User Group (CBI 130), Charles Babbage Institute, University of Minnesota, Minneapolis.

[66] Martin, J. *Computer Data-Base Organization*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1977.

[67] McCaffrey, J.J. *From Punched Cards to Personal Computers*, June 10 1989, John J. McCaffrey Memoirs (CBI 47), Charles Babbage Institute, University of Minnesota, Minneapolis.

[68] McClellan, S.T. *The Coming Computer Industry Shakeout: Winners, Losers, and Survivors*. John Wiley & Sons, New York, 1984.

[69] McCracken, D.D., Weiss, H. and Lee, T.-h. *Programming Business Computers*. John Wiley and Sons, New York, 1959.

[70] McGee, R.C. "Preliminary Manual for the Generalized File Maintenance System, SSD 046," December 23 1958, SHARE Records, 1955-86, NMAH 567, Archives Center, National Museum of American History, Behring Center, Smithsonian Institution, Washington, DC.

[71] McGee, R.C. The Structure of a Standard File, SSD 045, November 10 1958, SHARE Records, 1955-86, NMAH 567, Archives Center, National Museum of American History, Behring Center, Smithsonian Institution, Washington, DC.

[72] McGee, R.C. and Tellier, H. A Re-Evaluation of Generalization. *Datamation*, *6*, 4 (July-August 1960), 25-29.

[73] McGee, W.C. Generalization: Key To Successful Electronic Data Processing. *Journal of the Association for Computing Machinery*, *6*, 1 (January 1959), 1-23.

[74] McJones, P. The 1995 SQL Reunion: People, Projects and Politics, August 20 1997 [cited 12th Feb 2000] http://www.mcjones.org/System_R/SQL_Reunion_95/sqlr95-System.html

[75] Michaels, A.S., Mittman, B. and Carson, C.R. A Comparison of the Relational and CODASYL Approaches to Data-Base Management. *ACM Computing Surveys*, *8*, 1 (March 1976), 125-151.

[76] Miller, E. Information Retrieval--1961. *Datamation*, *7*, 10 (October 1961), 19-21.

[77] National Research Council *Funding A Revolution: Government Support for Computing Research*. National Academy Press, Washington, DC, 1999.

[78] Nolan, R.L. Computer Data Bases: The Future is Now. *Harvard Business Review*, *51*, 5 (September-October 1973), 98-114.

[79] Nolan, R.L. (ed.), *Managing the Data Resource Function*. West Publishing Co, New York, 1974.

[80] Olle, T.W. Recent CODASYL Reports on Data Base Management. In Rustin, R. ed. *Data Base Systems*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1972, 175-184.

[81] Osborn, R.F. GE and UNIVAC: Harnessing the High-Speed Computer. *Harvard Business Review*, *32*, 4 (July-August 1954), 99-107.

[82] Pansophic Systems Incorporated Competitive Products Survey: Data Base Management Systems Software Products, December 31 1981, Author's personal collection (donated by Joseph A Piscopo to Charles Babbage Institute, University of Minnesota).

[83] Parker, R.W. The SABRE System. *Datamation*, *11*, 9 (September 1965), 49-52.

[84] Patrick, R.L. Oral History Interview with Thomas Haigh, February 16 2006, Mountain View, CA, 2006, Forthcoming from Computer History Museum, Mountain View, CA.

[85] Performance Development Corporation An Interview With Charles W Bachman (Part II), Data Base Newsletter, Volume 8, No. 5, September 1980, Charles W. Bachman Papers (CBI 125), Charles Babbage Institute, University of Minnesota, Minneapolis.

[86] Pickard, A. The Status of GIS. In *SHARE XXXI Proceedings*, 1968, 2.27-22.43.

[87] Piscopo, J.A. "Oral History Interview by Thomas Haigh, 03 May, Washington DC," 2002, OH 342, Charles Babbage Institute, University of Minnesota, Minneapolis.

[88] Postley, J.A. Mark IV: Evolution of the Software Product, a Memoir. *IEEE Annals of the History of Computing*, *20*, 1 (January-March 1998), 43-50.

[89] Postley, J.A. and Jackobsohn, H. The Third Generation Computer Language: Parameters do the Programming Job. In Data Processing Management Association ed. *Data Processing, vol. 11*, Los Angeles, California, 1966, 408-415.

[90] Powers, V. Implementing Generalized Data Base Management Systems. *Data Management*, *13*, 5 (May 1975), 14-16.

[91] Pugh, E.W., Johnson, L.R. and Palmer, J.H. *IBM's 360 and Early 370 Systems*. MIT Press, Cambridge, MA, 1991.

[92] Reside, K.D. and Seiter, T.J. The Evolution of an Integrated Data Base. *Datamation*, *20*, 9 (September 1974), 57-60.

[93] Romberg, B.W. Data Bases: There Really Is a Better Way to Manage Your Files. *Infosystems*, *20*, 5 (May 1973), 56-57, 58.

[94] Rowan, T.C. The Recruiting and Training of Programmers. *Datamation*, *4*, 3 (May-June 1958), 16-18.

[95] Schubert, R.F. Basic Concepts in Data Base Management. *Datamation*, *18*, 7 (July 1972), 42-47.

[96] Schubert, R.F. Directions in Data Base Management Technology. *Datamation*, *20*, 9 (September 1974), 48-51.

[97] Schussel, G. When Not To Use a Data Base. *Datamation*, *21*, 11 (November 1975), 82, 91, 98.

[98] Shannon, C.E. and Weaver, W. *The mathematical theory of communication*. University of Illinois Press, Urbana, 1949.

[99] Simon, L. and Sisson, R. Evolution of a Total System. *Total Systems Letter*, *1*, 11 (January 1966), 1-4.

[100] Statland, N. and Hillegass, J.R. Random Access Storage Devices. *Datamation*, *9*, 12 (December 1963), 34-45.

[101] Steig, D.B. File Management Systems Revisited. *Datamation*, *18*, 10 (October 1972), 48-51.

[102] Stone, M.D. Data Processing and the Management Information System: A Realistic Evaluation of Data Processing's Role in the Modern Business Enterprise. In American Management Association ed. *Data Processing Today: A Progress Report -- New Concepts, Techniques and Applications -- AMA Management Report Number 46*. American Management Association, Finance Division, New York, 1960, 14-22.

[103] Synnott, W.R. and Gruber, W.H. *Information Resource Management: Opportunities and Strategies for the 1980s*. John Wiley & Sons, New York, 1981.

[104] System Development Corporation *Preprint for Second Symposium on Computer-Centered Data Base Systems*, Sponsored by SDC, ARPA, and ESD, September 1 1965, Burroughs Corporation Records (CBI 90), Charles Babbage Institute, University of Minnesota, Minneapolis.

[105] The Diebold Group, I. *Current Status of Data Bases and Data Base Management*, The Diebold Group, Cambridge, Mass, 1976.

[106] Various. Panel Discussion: SIGBDP 3 -- Performance Measurement and Data Base Design. In Association for Computing Machinery ed. *Proceedings of the 1975 Annual Conference*. Association for Computing Machinery, New York, 1975.

[107] Vorhaus, A.H. TDMS: A New Approach to Data Management. *Systems & Procedures Journal*, *18*, 4 (July-August 1967), 32-35.

[108] Waites, W.G. MIS or IMS? *Journal of Systems Management*, *22*, 1 (January 1971), 32-34.

[109] Webster, E. and Statland, N. Instant Data Processing. *Business Automation*, *7*, 6 (June 1962), 34-36, 38.

[110] Weindling, R.E. Office Will Run Every Business Activity. *Office Management and American Business*, *22*, 1 (January 1961), 12-15.

[111] Welke, L. A Review of File Management Systems. *Datamation*, *18*, 10 (October 1972), 52-54.

[112] Wellisch, H. From Information Science to Informatics: A Terminological Investigation. *Journal of Librarianship*, *4*, 3 (July 1972), 157-187.

[113] Wendler, C.C. What are the Earmarks of Effective Total Systems? *Systems & Procedures Journal*, *17*, 4 (July-August 1966), 29-31.

[114] Zand, M., Collins, V. and Caviness, D. A Survey of Current Object-oriented Databases. *ACM SIGMIS Database*, *26*, 1 (February 1995), 14-29.

Report from the First and Second International Workshops on Information Quality in Information Systems - IQIS 2004 and IQIS 2005 in conjunction with ACM SIGMOD/PODS Conferences

Monica Scannapieco
Università di Roma La Sapienza, Italy
monscan@dis.uniroma1.it

Laure Berti-Équille
IRISA, University of Rennes I France
berti@irisa.fr

ABSTRACT

This report summarizes the constructive discussions of the first two editions of the International Workshop on Information Quality in Information Systems, IQIS 2004 and IQIS 2005, held respectively in Paris, France, on June 13, 2004 and in Baltimore, MD, USA, on June 17, 2005.

1. WORKSHOP SCOPE

The problem of poor data quality stored in database-backed information systems is widespread in the governmental, commercial, and industrial environments.

Alarming situations with various information quality problems cannot be ignored anymore and, theoretical as well as pragmatic approaches are urgently needed to be proposed and validated. As a consequence, information quality is now becoming one of the hot topics of emerging interest in the academic and industrial communities.

Many processes and applications (such as information system integration, information retrieval, and knowledge discovery from databases) require various forms of data preparation or repair with several data processing and cleaning techniques, because the data input to the application-dedicated algorithms is assumed to conform to "nice" data distributions, containing no missing, inconsistent or incorrect values.

This leaves a large gap between the available "dirty" data and the available machinery to process the data for application-specific purposes.

The first and second editions of the International Workshop on Information Quality in Information Systems (IQIS), in conjunction with ACM SIGMOD/PODS conferences 2004 and 2005, focused on database-centric issues in data quality.

2. WORKSHOP PROGRAM

The main goal of the IQIS workshops is to be a forum for researchers, engineers, students, and practitioners from the database, knowledge discovery and data mining, information system engineering as well as statistics communities that have an in-depth interest in information quality in database-backed information systems, and also in the various techniques of data preparation, detection of inconsistent, contradictory or improbable data, data cleaning, and in ETL systems. The participants of the two editions of the workshop were invited to cover practical and theoretical issues of data quality. In the course of each workshop, various new approaches were presented to tackle the various problems of data quality in different application domains. In addition, the limitations and shortcomings of current approaches were discussed, and possible directions for future research in the domain were outlined. Areas of interest of these one-day events included:

- Metrics for information and data quality
- Quality-aware query languages and query processing
- Quality-aware integration
- Detection of outliers, duplicates, and inconsistencies
- Entity resolution, record and data linkage
- Intelligent data preparation and data cleaning
- Models, methodologies, and frameworks for information quality
- Application-driven information quality: bioinformatics, CRM, scientific applications
- Data type-dependent information quality: Web, multimedia, XML data.

2.1 IQIS 2004 Discussions

The topics for the technical sessions of the 2004 edition (see IQIS 2004 web site at http://www.hiqiq.de/iqis/) were structured in three sessions covering three relevant areas of the information quality research: data transformation and duplicate detection, assessment, and quality-driven information integration.

The keynote speech "Quality-Aware Data Integration in Peer-to-Peer Systems" was given by Maurizio Lenzerini (Università di Roma La Sapienza). He emphasized the importance of information quality in peer-to-peer system [14]. In these systems, every peer acts as both client and server, and provides part of the overall information available from a distributed environment, without relying on a single global view. He identified the research issues of data integration

in peer-to-peer systems, and outlined the impact of the notion of peer quality on both the semantics of the data integration systems, and the algorithms for query processing. Interesting discussions started because recent research has highlighted the importance of data quality issues (e.g., freshness [3]) in environments characterized by extensive data replication. Query processing in P2P environments has to include a record matching activity designed to exploit the presence of multiple overlapping sets of data. Moreover, complex data transformations for solving schema and data heterogeneities have to be applied [6]. Duplicated copies of the same data have to be compared in order to select and construct a best quality copy which is then returned for the global query and also submitted to the organizations having provided low quality copies of the same data.

In the thread of the discussions, performing duplicate detection on complex data appeared to be one of the main issues of the area discussed in this workshop.

Two papers on duplicate detection were then presented: the first one regarding XML duplicate detection [22] and the second one proposed a cost optimal decision model that takes into account the cost of erroneous classifications when deciding if two records are duplicates or not [21]. The dual problem of efficiently detecting patterns of conflicts in a pair of overlapping data sources has also been advocated by [19] and [18].

As the second theme of the workshop, the implications of information quality in data integration were considered with a specific focus on dealing with inconsistent data at query processing time. In [9], the authors proposed the definition of a formal semantics in a Global-As-View (GAV) data integration system when data retrieved at sources do not satisfy constraints on the global schema. A generalized framework for query answering in presence of inconsistent data sources has also been proposed in [15]. Finally, the results concerning information quality assessment including a methodology for data quality assessment from the user perspective [5] were presented.

2.2 IQIS 2005 Discussions

The topics for the technical sessions of the 2005 edition (see IQIS 2005 web site at http://iqis.irisa.fr) were structured in three main sessions covering: data quality models, record linkage, and statistics and clustering for ensuring data quality.

IQIS 2005 invited two keynote speakers. The first keynote speech "Handling Data Quality in Entity Resolution" was given by Hector Garcia-Molina (Stanford University). He discussed the challenges of the entity resolution problem under uncertain data (*i.e.*, the identification problem of matching records from multiple information sources that correspond to the same real-world entity) [10]. The second keynote speech "Methods and Analyses for Determining Quality" was given by William E. Winkler (U.S. Bureau of the Census, Statistical Research)[14]. He described situations where properly chosen metrics may indicate that data quality is not sufficiently high for monitoring processes, for modeling, and for data mining. Additionally, he described generalized methods that allow a skilled individual to perform massive clean-up of files in some situations.

Following the discussions first raised by Hector Garcia-Molina on the various issues of entity resolution, the main themes of IQIS 2005 workshop were centered on data linkage and data cleaning. Interesting exchanges between Dongwon Lee and Sharad Mehrotra started on the complementarity of their respective approaches for data linkage: the first one [13] presented a sampling-based approximate join algorithm considering the main problems commonly occuring in large-scale bibliographic digital libraries with *mixed citations* of different but homonymic scholars and *split citations* of the same author appearing under different name variants; the approach of [7] for object consolidation analyzed not only object features, but also additional information such as inter-object relationships.

Joining the discussion, the authors of [12] proposed two error-tolerant measures for identifying related attributes across independent databases to integrate based on similarities in their data.

The other important issue raised by record linkage that was discussed in the workshop concerns the privacy: in [2], the authors proposed a secure blocking scheme to improve significantly the performance of record linkage techniques while being secure.

The discussions were completed and focused on data cleaning: [8] proposed a data cleaning approach, based on modeling data dependencies with Markov networks. Belief propagation is used to compute the marginal and posterior probabilities, so as to infer missing values or to correct errors. In [11], the authors proposed a new framework for implementing active data warehousing with ETL activities over queue networks with minimal overhead and smooth upgrade of the data source systems.

In the session dedicated to data quality models, the authors of [17] proposed a model for formal data quality agreements between data providers and data consumers and they described an algorithm for dealing with constraints on the completeness of a query result with respect to a reference data source. Applied to the biological domain, [16] extended the semi-structured data model to include quality metadata with computing and updating useful data quality measures. In [20], the authors described a technique to rank data sources by characterizing data sources that agree with accurate or high-quality data sources as likely accurate.

In complex distributed systems (such as electronic payments processing systems), the authors of [4] proposed a framework that integrates three classes of models for detecting statistically significant changes from baselines, for explaining the reason of change occurences and for preventing data quality problems. In the last session dedicated to statistics and clustering techniques, [1] presented an algorithm for clustering mixed numerical and categorical data sets with associated confidence values to represent the certainty of correctness of the categorical values. An interesting discussion initiated between the academics and practitioners (VISA, Census, AT&T, etc.) concerned the need of benchmarks and massive sets of "real and dirty" data to test and compare the approaches.

For details of the papers and keynote speeches including slides from all presentations, please visit the workshop websites. The IQIS 2005 proceedings appear in the ACM Digital Library.

3. FUTURE PLANS FOR IQIS

The approaches discussed in the workshops were mostly based on academic research. As the workshop participants agreed on this argument, to evaluate the real value of the

approaches, they need to be tested and applied on real world applications on very large data sets. It was further remarked that cooperation between academia and industry is important, if not essential, for further development and testing of the approaches with practical data benchmarks dedicated to specific data quality problems (such as record linkage or duplicate detection). The two editions of IQIS workshop achieved their goal to provide a forum for interactive discussions. On several occasions during each one-day workshop, items for a common research agenda were debated, and enthusiastic feedbacks have been collected from the participants. Plans for future collaborations were discussed. Moreover, the third edition of the IQIS workshop will be organized on June 30th in conjunction with ACM SIGMOD/PODS 2006 conference in Chicago, IL, USA (see IQIS 2006 web site at http://queens.db.toronto.edu/iqis2006/).

4. ACKNOWLEDGMENTS

We would like to thank the program committee members, the keynote speakers, the authors of all submitted papers, and all the workshop participants. Special thanks are given to AT&T Labs Research and to the Università di Roma La Sapienza for their support.

5. ADDITIONAL AUTHORS

Additional authors: Felix Naumann (Humboldt-Universität zu Berlin, Germany, email: `naumann@informatik.hu-berlin.de`), co-chair of IQIS 2004, Carlo Batini (Università di Milano 'Bicocca', Italy, email: `batini@disco.unimib.it`) and Divesh Srivastava (AT&T Labs Research, Florham, NJ, USA, email: `divesh@research.att.com`), co-chairs of IQIS 2005.

6. REFERENCES

[1] Bill Andreopoulos, Aijun An, and Xiaogang Wang, Clustering Mixed Numerical and Low Quality Categorical Data: Significance Metrics on a Yeast Example, *IQIS 2005*.

[2] Ali Al-Lawati, Dongwon Lee, and Patrick McDaniel, Blocking-Aware Private Record Linkage, *IQIS 2005*.

[3] Mokrane Bouzeghoub and Verònika Peralta, A Framework for Analysis of Data Freshness, *IQIS 2004*.

[4] Joseph Bugajski, Robert L. Grossman, Eric Sumner, and Zhao Tang, An Event Based Framework for Improving Information Quality That Integrates Baseline Models, Causal Models and Formal Reference Models, *IQIS 2005*.

[5] Cinzia Cappiello, Chiara Francalanci, and Barbara Pernici, Data Quality Assessment from the User's Perspective, *IQIS 2004*.

[6] Paulo Carreira and Helena Galhardas, Execution of Data Mappers, *IQIS 2004*.

[7] Zhaoqi Chen, Dmitri V. Kalashnikov, and Sharad Mehrotra, Exploiting Relationships for Object Consolidation, *IQIS 2005*.

[8] Fang Chu, Yizhou Wang, D. Stott Parker, and Carlo Zaniolo, Data Cleaning Using Belief Propagation, *IQIS 2005*.

[9] Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati, Tackling Inconsistencies in Data Integration through Source Preferences, *IQIS 2004*.

[10] Hector Garcia-Molina, Handling Data Quality in Entity Resolution, *IQIS 2005*.

[11] Alexandros Karakasidis, Panos Vassiliadis, and Evaggelia Pitoura, ETL Queues for Active Data Warehousing, *IQIS 2005*.

[12] Andreas Koeller, and Vinay Keelara, Approximate Matching of Textual Domain Attributes for Information Source Integration, *IQIS 2005*.

[13] Dongwon Lee, Byung-Won On, Jaewoo Kang, and Sanghyun Park, Effective and Scalable Solutions for Mixed and Split Citation Problems in Digital Libraries, *IQIS 2005*.

[14] Maurizio Lenzerini, Quality-Aware Data Integration in Peer-to-Peer Systems, *IQIS 2004*.

[15] Zoran Majkic, General Framework for Query Answering in Data Quality Cooperative Information Systems, *IQIS 2004*.

[16] Alexandra Martinez, and Joachim Hammer, Making Quality Count in Biological Data Sources, *IQIS 2005*.

[17] Paolo Missier, and Suzanne Embury, Provider Issues in Quality-Constrained Data Provisioning, *IQIS 2005*.

[18] Amihai Motro, Philipp Anokhin, and Aybar C. Acar, Utility-based Resolution of Data Inconsistencies, *IQIS 2004*.

[19] Heiko Müller, Ulf Leser, and Johann-Christoph Freytag, Mining for Patterns in Contradictory Data, *IQIS 2004*.

[20] Raymond K. Pon, and Alfonso F. Cardenas, Data Quality Inference, *IQIS 2005*.

[21] Vassilios S. Verykios, and George V. Moustakides, A Generalized Cost Optimal Decision Model for Record Matching, *IQIS 2004*.

[22] Melanie Weis and Felix Naumann, Detecting Duplicate Objects in XML Documents, *IQIS 2004*.

[23] William E. Winkler, Methods and Analyses for Determining Quality, *IQIS 2005*.

Report on the 7th Workshop on Distributed Data and Structures (WDAS 2006)

Thomas Schwarz, S.J.
Department of Computer Engineering
Santa Clara University
Santa Clara, CA 95053, USA
tjschwarz@scu.edu

Mark Manasse
Microsoft Research, Silicon Valley
1065 La Avenida
Mountain View, CA, 94043, USA
manasse@microsoft.com

The seventh Workshop on Distributed Data and Structures (WDAS) took place on the campus of Santa Clara University on January 4 and 5, 2006 and drew participants actively working in research on distributed data, structures, and their applications. WDAS aims to stimulate the exchange of ideas and to be a forum for work in progress. The electronic version of the Proceedings contains all papers accepted at the conference [WDAS]. In addition, the workshop presentations helped to select a subset that will appear revised in the Proceedings in Informatics Series at Carleton Scientific. Two papers could not be presented by authors because of the difficulties of obtaining an entry visa to the United States, unfortunately a more and more common phenomenon. We now give an overview of the topics discussed at the workshop and the related presentations.

1 Distributed Storage Systems

The workshop had two keynote addresses. Darrell Long (UC Santa Cruz) laid out the reliability challenges facing large-scale storage systems. Institutions like those at the national laboratories now face demands for petabytes of highly available data using commodity disk drives. Once the installation reaches a certain scale, device loss and malfunction become normal occurrences and simple strategies such as RAID Level 1 or 5 are no longer sufficient to make data loss a rare occurrence. The challenge of better manageability needs to extend to failure detection and failure recovery. The resulting research problems need to be addressed by both academic research with its quest for innovation and by commercial research with its financial resources and product orientation.

Bisson, Wu, and Brandt deal with another problem of very large storage sites, namely energy consumption. The disk based storage system of the future might try to turn disks off as much as possible. As a trend, disks become smarter and the object storage devices of the near future will combine a small CPU with a network card and a disk drive and possibly a small amount of non-volatile memory such as flash or MRAM. Such a disk can react to its environment while the disk itself is powered off. Ceph, the object storage device file system project at UC Santa Cruz exploits these features. In the paper presented at the workshop, the authors improve of the state of the art by including knowledge of the power status of other disks into the decision process and by redirecting write requests from devices in low power mode.

Currently, metadata is not used extensively across applications, though individual applications that manage large music or photo collections maintain a rich set of metadata specific to the type of data and the application itself. Another project at the Storage Systems Research Center at UC Santa Cruz called *Graffiti* (Bobb, Eads, Storer, Brandt, Miller, Maltzahn) tries to investigate ways to make metadata more portable and more exploitable by adding a new distributed metadata layer above the operating system. In essence, a new metadata layer will make the task of application programmers simpler and allow sharing of metadata across three dimensions: applications, people, and computers.

2 Video-On-Demand

Video on demand system have high bandwidth requirements. Proactive or broadcasting protocols schedule streams at certain times and reduce bandwidth by sending the same video stream to more than one user. Reactive protocols use techniques such as batching and piggybacking; they can also force users to participate by sending a just received stream to other users (tapping or chaining). Pâris presented such a protocol that allows for different capacities between upload and download and maximizes the contribution of near-simultaneous consumers of the same video.

3 Traditional Data Structures

Otoo presented work on an extendible array data structure that was motivated by needs in scientific computing to change the range of a dimension of a multi-dimensional array while maintaining good performance.

Erlingson, Manasse, and McSherry use multiple hash functions to improve traditional, static hashing (an improvement on "Cuckoo Hashing" by Pagh and Rodler, Fotakis *et al.* and Panigrahy). Their technique can achieve load factors of 99.9% in 99% of all instances. This paper generated the longest discussions on how best to apply its techniques to distributed systems. One of the exciting implications of this work is the possibility of types of distributed hash tables using overlay networks. One should be able also to optimize the mappings of the bucket structures of distributed hash tables.

4 P2P

P2P databases or P2P information retrieval need to partition a large set of nodes into k partitions with additional requirements such as load-balancing without global knowledge. Bickson, Dolev, Weiss, Aberer and Hauswirth apply probabilistic graph models in the context of P2P systems for two problems. First, they count the number of nodes of a specific type without requiring a specific communication topology. Second, they perform a distributed graph coloring. Both algorithms are implemented using a belief propagation inference algorithm and exhibit astonishingly good accuracy and low overhead.

Martins, Pacitti, and Valduriez address the problem of replica consistency under more dynamic conditions on semantic reconciliation than the classical ones. These conditions prevail in P2P, grid and mobile computing systems. Users might perform updates, join, and leave the network whenever they wish. Existing semantic reconciliation solutions are typically performed at a single node and are inappropriate for such a dynamic environment. Starting from IceCube, the authors propose DSR, which enables optimistic multi-master replication and assures eventual consistency among replicas.

5 SDDS

Scalable Distributed Data Structures allow access to distributed data in times that are largely or even completely (LH*) independent of the number of storage sites and that do not require a central coordinator for data addressing. Introduction of dynamic data structures such as linear hashing and B-trees did away with static techniques such as static hashing and ISAM. As the industry starts dealing with large database tables that need to be distributed over several sites, these distributed tables are still administered in a static fashion. The Ph.D. work of Soror Sahri under Litwin presented a prototype called SD-SQL server that is directed towards remedying this situation. It uses SDDS principles to distribute a growing table without administrator intervention over more and more sites (currently up to 250). The user manipulates the tables through a dynamically adjusted and updatable distributed partitioned view. Queries are processed in parallel at all storage sites and response time benefits from the distribution.

The Ph.D. work of Damian Cieslicki under Schwarz is devoted to providing growing SDDS with a sub-layer providing scalable high availability in a generic manner. Availability is achieved by using erasure correcting codes (m/n codes) to create parity data with which data on lost nodes can be reconstructed. His work should improve considerably the update performance of LH*$_{RS}$ (Litwin *et al.,* TODS, Sept. 2005).

6 Service Oriented Computing

Gavran, Milanović, and Srblić deal with the interesting problem of developing large-scale distributed applications based on Service-Oriented Computing. They developed a "simple service composition" programming language. It allows for fast and simple design and implementation of applications in the service-oriented programming model.

7 Web Spam

Marc Najork from Microsoft Research – Silicon Valley gave a second keynote address in which he discussed heuristics for detecting spam web pages, a burden imposed on any search engine by manipulating content providers. His interesting presentation was based on his joint work with Fetterly and Manasse [FMN04].

Acknowledgments

The conference was sponsored by the Microsoft Research Silicon Valley (Mountain View, CA) and by the host, the School of Engineering at Santa Clara University. C. Maltzahn, W. Litwin, J. F. Paris, and P. Valduriez actively helped with this write-up.

Papers at WDAS 2006

J.F. Pâris: Using available client bandwidth to reduce the distribution costs of video-on-demand services.

Danny Bickson, Danny Dolev, Yair Weiss, Karl Aberer, Manfred Hauswirth: Indexing data-oriented overlay networks under belief propagation.

Ivan Gavran, Andro Milanović, Siniša Srblić: End-user programming languages for semantic reconciliation.

Vidal Martins, Esther Pacitti, Patrick Valduriez: A dynamic distributed algorithm for semantic reconciliation.

Ulfar Erlingson, Mark Manasse, Frank McSherry: A cool and practical alternative to traditional hash tables.

Timothy Bisson, Joel Wu, Scott Brandt: A distributed spin down algorithm for an object-based storage device with write redirection.

Witold Litwin, Soror Sahri, Thomas Schwarz: Prototyping SD-SQL server: a scalable distributed database system.

Damian Cieslicki, Stefan Schäckeler, Thomas Schwarz: Highly Available Distributed RAM (HADRAM): Scalable availability for scalable distributed data structures.

Nikhil Bobb, Damian Eads, Mark Storer, Scott Brandt, Carlos Maltzahn, Ethan Miller: Graffitti: A framework for testing collaborative distributed metadata.

Syed Ahsan and Abad Shah: Biological databanks: distribution, heterogeneity, diversity, and provenance.

Ekow Otoo: Parallel and distributed access of dense multi-dimensional extendible array files.

References

[FMN04] Dennis Fetterly, Mark Manasse, and Marc Najork. Spam, Damn Spam, and Statistics: Using Statistical Analysis to Locate Spam Web Pages. 7th International Workshop on the Web and Databases (June 2004)

[WDAS] The WDAS proceedings are available in electronic form at www/soe.ucsc.edu/wdas06.

Report on the 2nd International Workshop on Data Integration in the Life Sciences (DILS'05)

Amarnath Gupta
gupta@sdsc.edu

Bertram Ludäscher
ludaesch@ucdavis.edu

Louiqa Raschid
louiqa@umiacs.umd.edu

Following a successful first international workshop on *Data Integration in the Life Sciences* (DILS) in 2004 in Leipzig, Germany [2], the second DILS workshop was held in San Diego, California from July 20–22, 2005 [1]. This new workshop series reflects the strong interest in an annual event, bringing together biologists and computer scientists conducting research in life science data management and integration, in domains and applications such as molecular biology, biodiversity, drug discovery and personalized medical research. The program committee accepted 15 long papers and 5 short papers from 42 submissions. In addition, DILS also featured 7 posters, 2 keynotes, several reports on ongoing research activities in academia and industry, and a panel on *The Electronic Health Record of the Future: Incorporating Molecular Information*, organized by the AMIA Genomics Working Group.

Setting the theme of the workshop, Shankar Subramaniam, Professor of Bioengineering and Chemistry at UCSD, noted in his keynote *Challenges in Biological Data Integration in the Post-Genome Sequence Era* that "the standard paradigm in biology that deals with '*hypothesis to experimentation (low throughput data) to models*' is being gradually replaced by '*data to hypothesis to models*' and '*experimentation to more data and models*'." Given the complexity and incompleteness of data in biology, he called for robust data repositories that allow interoperable navigation, query and analysis across diverse data, a plug-and-play environment that will facilitate seamless interplay of tools and data and versatile biologist-friendly user interfaces. In the second keynote, *Curated Databases*, Peter Buneman noted that biological data is often created and maintained at a high cost involving extensive human curation, and he explored the relationship between database research and data curation. He identified challenges in annotation, provenance, archiving, publishing and security and emphasized the goal of making sure that curated data is accessible and understandable to future biologists.

The twenty accepted research papers covered a wide spectrum of theoretical and practical research issues including user interfaces, analysis tools, scientific/clinical workflows, ontologies, and data integration techniques. Below we highlight some of the contributions.

User Interfaces and Analysis Tools. S. Cohen-Boulakia, S. Davidson and C. Froidevaux describe *A User-Centric Framework for Accessing Biological Sources and Tools*, based on an analysis of scientists' needs. They presented the BioGuide system that helps biologists choose among a variety of resources, taking into account *queries*, *preferences* and *strategies*. Information resources are connected in a graph; then information requests are processed by finding appropriate paths in this graph using the preferences and strategies. The research studies the semantics of this user-centric framework and its complexity.

The *Hybrid Integration of Molecular-Biological Annotation Data* from public sources is studied by T. Kirsten, H.-H. Do, C. Körner, and E. Rahm. Annotation data is integrated in a virtual (*i.e.*, mediator) approach, coupling SRS with a warehouse of expression data. Their system exploits correspondences between molecular-biological objects in the integration and supports functional analysis using annotations from GeneOntology, Locuslink and Ensembl.

The *BioNavigation* system by Z. Lacroix *et al.* allows scientists to express queries over semantic concepts and labeled relationships. Their ESearch algorithm generates all possible source paths following links and paths through data resources, and then ranks these source paths using source metadata metrics such as the number of result objects in the target, or the evaluation cost of a source path.

N. Tran *et al.* present a BioSigNet-RR, a *Knowledge-Based Integrative Framework for Hypothesis Formation in Biochemical Networks*. Hypothesis formation is modeled as a reasoning process that attempts to explain all concepts that are not completely entailed by the current knowledge of the system, by logically extending the knowledge base. It has been applied to the problem of modeling typically incomplete knowledge about biochemical networks.

In *BioLog: A Browser Based Collaboration and Resource Navigation Assistant*, P. Singh *et al.* describe a system for biomedical researchers that archives access patterns of scientists as they browse PubMed and also allows users to browse group specific archives. It makes recommendations using gene-to-gene, abstract-to-abstract and user-to-user relevance networks that use a combination of collaborative filtering and content based filtering techniques.

Challenging Applications. O. Fiehn *et al.* describe a system for *Setup and Annotation of Metabolomic Experiments*, noting that long-term reusability of metabolomic data needs correct metabolite annotation and consistent biological classification, and that "data without metadata are junk". The system attempts to produce automatic annotation of the results of mass spectrometry experiments, and feeds the annotated information back to a LIMS system for verification and use by scientists.

J. Kang *et al.* address the problem of *Integrating Heterogeneous Microarray Data Sources using Correlation Signatures*. The paper develops a technique to compute the *signature vector* of a gene with other genes in a microarray experiment, and then perform a statistical correlation among these signatures. They also propose an OLAP-inspired structure called the *gene signature cube* to perform global analysis of multiple experiments.

S. Jablonski *et al.* report on *Building a Generic Platform for Medical Screening Applications based on Domain Specific Modeling and Process Orientation*. They propose a process-oriented approach and system for distributed screenings for the early detection and diagnosis of glaucoma disease.

Tools for Legacy Data and Data Integration Solutions. Wrappers are crucial to the task of accessing legacy biological data sources and automatic wrapper generation is an ongoing challenge. K. Sinha *et al.* report on a tool to learn the layout and generate wrappers for flat-file datasets. The tool uses a number of heuristics to determine file delimiters, semiautomatically eliminate incorrectly formulated files and create a parser for the data files. The authors report on experiments with Swissprot, Genbank and Pfam data.

Associating a unique key with an object is crucial to effective data management. G. Neglur *et al.* report on applying the "Unique SMILES" notation to chemical structures and show that the prevalent algorithm relies on symmetry properties of graphs and will fail for certain graphs. Their paper on *Assigning Unique Keys to Chemical Compounds for Data Integration* modifies the algorithm by introducing a step where a "tie-breaking" is performed for formulas having the same tree-expansion. They demonstrate that it creates unique SMILE codes for all their counterexamples.

E. Guérin *et al.* describe GEDAW, an object-oriented gene expression data warehouse that integrates public data on expressed genes, experiment data from microarrays and other relevant data resources, *e.g.*, ontologies such as GO and UMLS. The objective is to combine biological mechanisms and medical knowledge with experiment data, and the project addresses issues in both semantic data integration and analysis of the integrated resources.

The AutoMed tool designed with a hypergraph data model for warehouse-based integration has been in existence for some time. M .Maibaum *et al.* discuss the adaptation of this system to biological data integration. Specifically, the paper discusses the issue of information integration when part of the data to be integrated are structured, and another part has to be clustered based on data values to find equivalent entities.

M. Mahoui *et al.* explore the concept of semantic correspondence for biological objects that are actually similar but are differently represented in their respective data sources. The paper puts forward the notions of *degree of semantic correspondence* and *cardinality of semantic correspondence* as two measures of such correspondence and shows how they can be used for information integration across heterogeneous sources.

There is an increasing number of internet-accessible services for providing, comparing and transforming biological data. A. Ngu *et al.* consider the problem of automatically classifying these sources based on their output and the kind of user interaction they need. They develop the notion of a *service class descriptors*, that captures metadata that would be adequate to describe these services, and present two techniques to automatically construct the service class descriptors, and discuss their applicability and performance.

Integrating private data with large public data banks can be expensive if they are to provide absolute privacy. R. Pon *et al.* provide a technique based on the semi-join operation; it exploits the uncertainty caused by hash collisions and injects noise. This method provides an upper bound on the amount of privacy loss during data integration.

Ontologies and Data Integration. As the number and sizes of ontologies increase, there is a growing need to query repositories of large ontologies. S. Trißl *et al.* consider the problem of querying ontologies that are structured like trees and DAGs, and offers indexing schemes for both cases. They analyze the performance of queries using these index structures for a number of common query patterns in biological ontologies.

Taxonomic identification is key to research in several domains. While biologists identify their data with scientific names, this is insufficient to unambiguously distinguish taxon concepts. A model for the representation of diverse taxonomic concepts is presented by J. Kennedy *et al.*

The INDUS system by D. Caragea *et al.* employs ontologies and inter-ontology mappings to provide a user view of multiple heterogeneous sources that reflects the user's preferred ontology. The authors present an experiment where INDUS is used to learn probabilistic models to predict GO functional classifications; mappings such as EC2GO and MIPS2GO are used in this task.

Ontologies are expected to have significant impact on research in ecology. STELLA is a widely adopted software tool for ecological modeling. C.M. Keet presents a formal mapping between STELLA and ontology elements. This

mapping simplifies ontology development and can support semi-automated techniques for ontology development.

P. Mork *et al*. describe *The Multiple Roles of Ontologies in the BioMediator Data Integration System*. Queries are expressed against concepts in the ontologies; at the same time they may serve as data sources. A system ontology provides metadata about sources, *e.g.*, how often it is updated. Finally, ontologies are used to describe the mapping from data sources to the BioMediator mediated schema.

DILS'06. The third international DILS workshop will be held at European Bioinformatics Institute (EBI) at Hixton, south of Cambridge, UK from July 20–22, 2006; for details see `www2.informatik.hu-berlin.de/dils2006`.

References

[1] B. Ludäscher and L. Raschid, editors. *Data Integration in the Life Sciences, Second InternationalWorkshop, DILS 2005, San Diego, CA, USA, July 20-22, 2005, Proceedings*, volume 3615 of *Lecture Notes in Computer Science*. Springer, 2005.

[2] E. Rahm, editor. *Data Integration in the Life Sciences, First International Workshop, DILS 2004, Leipzig, Germany, March 25-26, 2004, Proceedings*, volume 2994 of *Lecture Notes in Computer Science*. Springer, 2004.

Report on the 7th ACM International Workshop on Web Information and Data Management (WIDM 2005)

Angela Bonifati
Icar CNR, Italy
bonifati@icar.cnr.it

Dongwon Lee
Penn State University, USA
dongwon@psu.edu

Abstract

In this report, to our best recollection, we provide a summary of the 7th ACM International Workshop on Web Information and Data Management (WIDM 2005), which took place at the Hilton Bremen Hotel, in Bremen, on November 5, 2005, in conjunction with the 14th ACM Int'l Conf. on Information and Knowledge Management (CIKM).

1 Workshop Overview

The WIDM [2] (pronounced like "Widom") was originally started by Fereidoon Sadri (University of North Carolina at Greensboro) and Cyrus Shahabi (USC) in late 90s, and has reached its 7th in its series (all held in conjunction with ACM CIKM 2005). Initially, WIDM was one of few workshops that bridge DB and IR community to the (then) emerging Web community. About the same time as WIDM in 1998, for instance, WebDB has also started with the similar research themes.

This year, WIDM implemented a one-day program, and attracted 44 submissions from 15 countries, making the selection very competitive. All the papers were reviewed by 3 members of the Program Committee. After a discussion phase on a few papers with conflicting reviews, 12 papers were accepted, among which 8 full papers and 4 short papers. This represents a highly competitive acceptance rate of 27%. The covered topics include among the others Web Mining, Web and XML Data Management, Semantic Web, Web Commerce, Advanced Web Applications, Web Exploration, and Performance of Web Applications. Submissions covered most of the above topics with a predominance of papers on Web ranking and retrieval, XML data management and Web discovery, Web clustering and filtering. These happen to be some of the "hottest" topics that the community has been working on these days.

2 Technical Program

The 12 accepted papers were divided into three technical sessions. The papers can be downloaded from the workshop website [1], which has also additional information about the program.

2.1 Session 1: Web Ranking and Retrieval

The first paper "*Web Path Recommendations Based on Page Ranking and Markov Models*" [4] tried to overcome the shortcomings of current techniques to predict users' navigational behavior. The authors proposed the use of a PageRank-style algorithm for assigning prior probabilities to the web pages based on their importance in the web site graph, and experimentally showed that their approach yielded more objective and representative predictions than the ones produced from the pure usage-based approaches.

The second paper "*Semantic Similarity Methods in WordNet and their Application to Information Retrieval on the Web*" [5] studied semantic similarity methods using WordNet. Despite the arrival of the Semantic Web era and a growing support of metadata, it is still unclear how to measure the "similarity" of two ontologies that bear little lexicographic similarity. In the paper, the authors investigated approaches that map terms (or concepts) to the ontology graph of WordNet and examined their relationships.

The last paper "*DirectoryRank: Bringing Order to Web Directories*" [6] presented DirectoryRank, a ranking framework that orders the web pages within a given topic according to how informative they are about the topic.

2.2 Session 2: XML Data Management and Web Discovery

In the first paper "*Exploiting Native XML Indexing Techniques for XML Retrieval in Relational Database Systems*" [7], the authors aim at reducing the gap between native XML database and XML-to-RDBMS systems. To that purpose, the authors examined how native XML indexing techniques can boost the retrieval of XML stored in an RDBMS, and proposed the Relational CADG (RCADG), an adaptation of several native indexing approaches to the relational model.

The second paper "*Query Translation Scheme for Heterogeneous XML Data Sources*" [8] investigated the query and data schema mismatching problem, and proposed an inclusion mapping algorithm that decides how compatible the schema of the query and that of the target XML documents are.

In the third paper "*Impact of XML Schema Evolution on Valid Documents*" [9], the authors studied the problem of the evolution of an XML Schema, and proposed to minimize document revalidation by detecting the document parts potentially invalidated by the schema changes.

The problem of discovering and composing the right web services from a pool of many is an important problem. In this context, the last paper "*A Framework for Semantic Web Services Discovery*" [10] proposed a framework for ontology-based flexible discovery of semantic web services. That is, their approach relied on user-supplied, context-specific mappings from an user ontology to relevant domain ontologies used to specify web services.

2.3 Session 3: Web Clustering, Filtering and Applications

The first paper "*Narrative Text Classification for Automatic Key Phrase Extraction in Web Document Corpora*" [11] investigated the significance of narrative text classification in the task of automatic key phrase extraction in Web document corpora. That is, according to their findings, key phrases extracted from the narrative text only are significantly better than those obtained from all plain text of Web pages.

The second paper "*On improving Local Website Search Using Web Server Traffic Logs: A preliminary Report*" [12] presented their study on the importance of web server traffic logs to improve "local" search, in addition to pure link counts as in PageRank.

The "shilling attack" can be defined as malicious user or set of users attempting to change the behavior of the system to suit their own needs. In the third paper "*Preventing Shilling Attacks in Online Recommender Systems*" [13], the authors proposed several metrics for analyzing rating patterns of malicious users, and suggested an algorithm for protecting recommender systems against shilling attacks.

The fourth paper "*Looking at Both the Present and the Past to Efficiently Update Replicas of Web Content*" [14] concerns the problem of keeping the copies of web pages in sync with their original copies. To solve this problem, they proposed a new approach that learns to predict the change behavior of web pages based both on the static features and change history of pages, and refreshes the copies accordingly.

Clustering the results of a search helps the user to overview the information returned. In the last paper of the workshop "*A Search Result Clustering Method using Informatively Named Entities*" [15], the authors regarded the clustering task as indexing the search results. Their solution first extracted named entities as labels, and used the importance in the search result and the relation between terms and search queries in selecting the right labels. In their prototype, the proposed solution showed higher performance than existing methods.

2.4 Keynote Address

This year's keynote address was given by Prof. Donald Kossmann from ETH Zurich, Switzerland with the title "*A Web of Data; New Architectures for New Technology?*" [3]. The development and evolution that the Web has undergone this last decade has not been coupled with an evolution of the way we build Web applications. Imperative programming languages (e.g., Java or C#) and middleware architectures are still dominant in industry. His talk discussed why these old architectures are problematic and spurred interesting ideas for novel software architectures to build Web applications. The keynote talk was immediately followed by several questions from the workshop audience on the major research issues behind this technology and its future directions. In particular, the talk highlighted the importance of new infrastructures for Web applications, and the emerging needs of several IT institutions. These are indeed experimenting the use of XML data in their applications, and in general of new cutting-edge technologies. An interesting issue emerged during the talk was the high flexibility of XQuery, the standard XML query language to be used as a full-fledged web programming language as well. Being Turing-complete, XQuery is unsurprisingly powerful and could in the long term be coupled with the Java technology in the web development process. To such

a purpose, the keynote speaker has advocated his teaching experience with XQuery courses, and the observed fast learning curve, not to mention his consulting experience within the major financial institutions in Switzerland.

3 Final Thoughts

WIDM 2005 was very successful. It proved the effectiveness of a one-long-day workshop with high quality of talks and papers resulted in a lively and interesting discussion carried through the entire workshop. The proceedings of the workshop have been published by ACM [2]. The next WIDM, the 8th one in the series, will take place in conjunction with ACM CIKM 2006, in Arlington (VA) on November 10, 2006.

Acknowledgements

WIDM 2005 would have not been successful without the contributions of all the members of the organizing committee and the technical program committee. WIDM 2005 was sponsored by ACM SIGIR in cooperation with ACM SIGMOD, and Geselschaft für Informatik e.V. (GI).

References

[1] WIDM 2005 workshop web site, WIDM 2005 Program, WIDM 2005 Organizing and Program Committees, November 2005. "URL: http://pike.psu.edu/widm05/".

[2] A. Bonifati, and D. Lee. (Eds.). "7th ACM Int'l Workshop on Web Information and Data Management (WIDM 2005)". Bremen, Germany, November 4, 2005. ISBN 1-59593-194-5

[3] D. Kossmann. "A web of data: new architectures for new technology". In Proc. of WIDM 2005, page 1.

[4] M. Eirinaki, and M. Vazirgiannis and D. Kapogiannis. "Web path recommendations based on page ranking and Markov models". In Proc. of WIDM 2005, pages 2-9.

[5] G. Varelas, and E. Voutsakis, and P. Raftopoulou, and E.G.M. Petrakis, and E.E. Milios: "Semantic similarity methods in wordNet and their application to information retrieval on the web". In Proc. of WIDM 2005, pages 10-16.

[6] V. Krikos, and S. Stamou, and P. Kokosis, and A. Ntoulas, and D. Christodoulakis. "DirectoryRank: ordering pages in web directories". In Proc. of WIDM 2005, pages 17-22.

[7] F. Weigel, and K.U. Schulz, and H. Meuss. "Exploiting native XML indexing techniques for XML retrieval in relational database systems". In Proc. of WIDM 2005, pages 23-30.

[8] C.X. Chen, and G.A. Mihaila, and S. Padmanabhan, and I. Rouvellou. "Query translation scheme for heterogeneous XML data sources". In Proc. of WIDM 2005, pages 31-38.

[9] G. Guerrini, and M. Mesiti, and D. Rossi. "Impact of XML schema evolution on valid documents". In Proc. of WIDM 2005, pages 39-44.

[10] J. Pathak, and N. Koul, and D. Caragea, and V. Honavar. "A framework for semantic web services discovery". In Proc. of WIDM 2005, pages 45-50.

[11] Y. Zhang, and A.N. Zincir-Heywood, and E.E. Milios. "Narrative text classification for automatic key phrase extraction in web document corpora". In Proc. of WIDM 2005, pages 51-58.

[12] Q. Cui, and A. Dekhtyar. "On improving local website search using web server traffic logs: a preliminary report". In Proc. of WIDM 2005, pages 59-66.

[13] P-A. Chirita, and W. Nejdl, and C. Zamfir. "Preventing shilling attacks in online recommender systems". In Proc. of WIDM 2005, pages 67-74.

[14] L. Barbosa, and A.C. Salgado, and F. de Carvalho, and J. Robin, and J. Freire. "Looking at both the present and the past to efficiently update replicas of web content". In Proc. of WIDM 2005, pages 75-80.

[15] H. Toda, and R. Kataoka. "A search result clustering method using informatively named entities". In Proc. of WIDM 2005, pages 81-86.

Data Management Research at the Knowledge and Database Systems Lab (NTU Athens)

Timos Sellis, Yannis Vassiliou
Computer Science Division
School of Electr. and Comp. Engineering
National Technical University of Athens
Athens, Greece
{timos,yv}@dblab.ece.ntua.gr

1. INTRODUCTION

The Knowledge and Database Systems Lab (KDBSL) of the Electrical and Computer Engineering Dept. in the National Technical University of Athens was founded in 1992 by Prof. Timos Sellis and Prof. Yannis Vassiliou. Its activities involve theoretical and applied research in the area of Databases and Information Systems. The lab employs three postdoc researchers (Dr Theodore Dalamagas, Dr Alkis Simitsis, Dr Yannis Stavrakas), several PhD students and many graduate students. It has been involved in many research projects supported by the EU, international institutions, Greek organizations, the Greek Government and industrial companies.

This report presents a brief description of the major research activities of KDBSL. Those activities involve (a) data stream management, (b) modelling and operational issues in databases and data warehouses and (c) Web and P2P data management. More details about the activities, as well as the full list of KDBSL publications, are available at the site of the lab: `http://www.dblab.ece.ntua.gr`.

2. DATA STREAM MANAGEMENT

Stream processing must keep up with the fluctuating arrival rate of high-volume data sets. Hence it is not expected that fast in-memory computation can be applied over the entire stream. We consider that approximate query processing over compact, precomputed data synopses is sometimes the only viable option. The reason is that users can often tolerate small imprecision in query results, as long as these results are quickly generated and accompanied with accuracy guarantees. Besides, window specifications can be used to limit the scope of processing items as a means of providing incremental response to continuous queries. Finally, our interest also involves the special case of trajectory streams generated by traces of objects moving in a multidimensional space.

2.1 Constructing Optimal Wavelet Synopses

The wavelet transform is a powerful tool for constructing concise synopses of massive multi-dimensional data sets and rapidly changing data streams, while at the same time providing fast, approximate answers with accuracy guarantees. This transformation results in a set of wavelet coefficients offering a multi-resolution view of the data. Yet it requires only linear computational time and space. Due to its hierarchical decomposition, only a handful of the produced coefficients suffice to achieve a good summary of the data. As a result, an effective synopsis can be obtained by keeping those important coefficients and explicitly setting others to zero.

The work in [1] provides efficient I/O methods to create and maintain wavelet synopses over large multidimensional data sets. Results and space-time trade-offs are also presented in case of multi-dimensional data streams, when one of the dimensions (e.g., the time dimension) is continuously growing. For the more general and demanding case of aggregating streaming data, sketching techniques (i.e., randomized linear projections) need to be employed. The work in [2] solves the problem of estimating the most significant coefficients on the fly, as updates are streaming at high rates. The suggested method relies on two key technical ideas: (i) work directly on the wavelet domain by translating stream items; and (ii) a novel stream synopsis, termed Group-Count Sketch, which can be employed over hierarchically grouped coefficients to enable quick identification of important coefficients using a binary-search-like technique.

Our ongoing research focuses on constructing optimal synopses for more meaningful error metrics to measure the synopsis approximation quality. Improving space utilization, e.g., by employing adaptive quantization of coefficient values, can lead to significantly more accurate wavelet synopses.

2.2 Window Semantics for Streams

Since the total size of a stream is unknown, *windows* have been suggested as a means of limiting the amount of data given for processing and, thus, providing real-time responses to *continuous queries*. In [3] we have developed a foundation with formal semantics for specifying windows over data streams, and proposed a careful taxonomy of the most significant variants (count-based and time-based sliding, partitioned, landmark, and tumbling windows).

On the basis of rigorous algebraic specifications, we attempt to integrate rich windowing constructs with principal relational operators (selection, projection, and join). Our aim is to gain expressiveness for a wide range of SPJ continuous queries over streams.

We have developed a prototype stream processing engine to verify the correctness of windowing semantics and to investigate perspectives of syntactic equivalences and query

optimization. Overall, we consider windows as first-class citizens in stream processing. Windows should be treated as an indispensable ingredient of a stream algebra and query language towards a stream-enabled SQL.

2.3 Managing Trajectory Streams

Positioning applications are becoming extremely popular, e.g., for sensor monitoring or location-based services, due to the recent advances in telecommunications and location-enabled facilities like GPS, RFIDs, PDAs etc. *Trajectory data* from tracing movement of people, vehicles, animals etc. is constantly collected from multiple sources. Then, data is transmitted to a central processor as continuous, time-varying and possibly unbounded data streams of timestamped locations. In [4] we introduce a model that captures (a) the multidimensional nature of trajectories, (b) the interaction between space and time, and (c) frequent positional updates and interrelationships of moving objects with stationary entities. In such a dynamic setting, we believe that a generalized notion of windowing operators that combine temporal and spatial properties would be appropriate. Such a notion can effectively capture the evolving characteristics of trajectories, especially with respect to *kinetic operations*.

Moreover, if approximate query answering is accepted as a trade-off to achieve real-time responses, then probably the amount of trajectory data should be reduced. In [5] we introduce two novel compression techniques based on *sampling* that take advantage of features pertinent to trajectories and yield reliable approximation quality at affordable computation overhead. The first technique is based on spatiotemporal thresholds that examine speed and orientation of moving objects. The second one attempts to preserve the shape of each trajectory, while maintaining a minimal sequence of point locations.

Moreover, we aim to exploit other synopsis techniques (like wavelets or sketches) for streaming trajectories, taking into account their multidimensional properties to efficiently support spatiotemporal continuous queries, such as range aggregates or similarity search.

3. MODELLING AND OPERATIONAL ISSUES IN DB'S AND DW'S

Research on modelling and operational issues of databases and data warehouses is another area of interest for our group. We study modelling problems such database schema evolution, pattern warehouses, the design and optimization of the internal processes of a data warehouse, and context-aware databases. We also study operational problems in databases. We propose a framework for a novel type of queries, precis queries, i.e. free-form queries that generate entire multi-relation databases. Such databases contain not only items directly related to the query selections but also items implicitly related to them. The query output may be transformed in narrative form providing much greater insight into the original data. Finally, our group also studies novel indexing schemes for containment queries.

3.1 Modelling & Optimization of ETL processes

Data warehouse operational processes normally compose a labor intensive workflow and constitute an integral part of the back-stage of data warehouse architectures. To deal with this workflow and to facilitate and manage the data warehouse operational processes, specialized processes are used under the general title *Extraction-Transformation-Loading (ETL) processes*. ETL processes are responsible for the extraction of data from several sources, their cleansing, their customization and transformation, and finally, their loading into a data warehouse.

We have studied the design, development and optimization of ETL processes [6]. The work in [7] proposes a novel *conceptual model* for the early stages of a data warehouse project. In those stages the time constraints of the project require a quick documentation of the involved data stores and their relationships rather than an in-depth description of a composite workflow.

Moreover, we have presented a formal *logical model* for the ETL environment. The model concentrates on the flow of data from the sources towards the data warehouse through the composition of transformations and data stores. It has two main characteristics: *genericity* and *customization* [8, 9]. The work in [9] also presents a palette of several templates to represent frequently used ETL activities along with their semantics and their interconnection. In [10] we build upon existing graph-based modelling techniques that treat ETL workflows as graphs [11]. We (a) extend the activity semantics to incorporate negation, aggregation and self-joins, (b) complement querying semantics with insertions, deletions and updates, and (c) transform the graph to allow zoom-in/out at multiple levels of abstraction. In [12] we exploit our modelling framework to introduce rigorous techniques to measure ETL workflows, and formally prove their applicability and correctness. We have complemented these two models with a method for the semi-automatic transition from the conceptual to the logical model for ETL processes [13].

Additionally, we have delved into the *logical optimization* of ETL processes to find the optimal ETL workflow [14]. We have modelled this problem as a state-space search problem. Each ETL workflow is considered as a state and the state space is fabricated through a set of correct state transitions [15]. Moreover, we provide algorithms towards the minimization of the execution cost of an ETL workflow.

Finally, we have implemented ARKTOS II, a prototype ETL tool to facilitate the design, the (re-)use, and the optimization of ETL workflows [16].

Our ongoing work involves the physical optimization of ETL workflows taking into account physical operators and access methods. Another challenge is to apply our findings in more general workflows (not just ETL ones). Future plans for ARKTOS II involve the extension of data sources to more sophisticated data formats, other that relational domain, like object-oriented or XML data.

3.2 Database Schema Evolution

Current database systems are continuously evolving environments, where design constructs are added, removed or updated rather often. Even trivial changes (e.g., addition or deletion of an attribute, modification of a constraint) in the schema of the database greatly affect a wide range of applications and queries built around it, which may become syntactically or semantically invalid. The problem of database schema evolution has been addressed under several contexts, e.g., OODBMS, RDBMS, XML. Nevertheless, problems persist, mainly due to the fact that in most cases, the proper

attention is not given to the role of queries as integral parts of the environment and therefore queries are not designed to handle database evolution. Our research introduces techniques to adapt queries and views to schema changes in the underlying database, in such way that the human interaction is minimized.

Specifically, our approach provides a mechanism to perform what-if analysis for potential changes of database configurations [17]. A graph model that uniformly models queries, views, relations and their significant properties (e.g., conditions) is introduced. Apart from the simple task of capturing the semantics of a database system, the graph model allows us to predict the impact of a change over the system. Furthermore, we provide a framework to annotate database constructs (including queries and views) with policies concerning their behavior in the presence of hypothetical changes occurring in the database schema. Rules that dictate the proper adaptation actions to *additions* or *deletions* performed to *relations*, *attributes* and *conditions* (all treated as first-class citizens of the model) are provided. Finally, a tool to visualize and performe what-if analysis for several evolution scenarios is being implemented [18].

3.3 Logical Subsets of Databases

The wide spread usage of database systems nowadays has brought the need to provide inexperienced users with the ability to easily search a database with no specific knowledge of a query language. Several recent research efforts have focused on supporting keyword-based searches over relational databases.

We have presented an alternative proposal and we have introduced the idea of *précis queries* [19]. These are free-form queries whose answer (a précis) is a synthesis of results, containing not only items directly related to the given query selections but also items implicitly related to them in various ways with the purpose of providing to the user much greater insight into the original data. Précis queries include two additional novelties: they do not generate individual relations but entire multi-relation databases, and query results are customized to a user or group of users and/or domain requirements [20].

Currently, we are working on the development of a précis query answering prototype, named PRÉCIS, with the following characteristics: (a) support of a keyword-based search interface for accessing the contents of the underlying database, (b) generation of a logical subset of the database that answers the query, which contains not only items directly related to the query selections but also items implicitly related to them in various ways, (c) customization of the logical subset generated and hence the précis returned according to the needs and preferences of the user as an individual or as a member of a group of users, and (d) translation of the structured output of a précis query into a narrative synthesis of results. This output is an English presentation of short factual information précis [21].

3.4 Indexing of Set-Valued Attributes

Containment queries on set-values emerge in a variety of application areas ranging from scientific databases to XML documents. Examples of set valued data can be found in market basket analysis, production models, image and molecular databases. Moreover, as RDBMS's and IR come closer (often in the interest of storing and handling XML and Web data), containment queries on set values become a use case of high significance for an RDBMS.

Despite the fact that set values are supported by the object relational model and by the most popular commercial RDBMS's, indexing techniques for set values and containment queries are limited. The basic alternatives for indexing set values still remain the inverted file and signature based methods, with the former offering superior performance in most cases.

We work on the design and implementation of novel indexing schemes that do not suffer from the drawbacks of the inverted file, and outperform it for containment queries [22]. Moreover, we investigate the evaluation of more complex queries on set values, involving ranking and similarity. We focus on proposing indices to support such queries, giving efficient IR functionality in an RDBMS.

3.5 Pattern Management

The vast volumes of information produced nowadays, makes the handling and storing of data increasingly harder. Still, the huge volumes of data do not constitute knowledge per se, but special extraction methods have to be employed in order to produce knowledge artefacts. We call these artefacts *patterns*. Till now, patterns had not been treated as first class citizens in most data models. The challenge from the database perspective is to develop models and tools for storing, querying and linking patterns with the underlying data.

Patterns are usually quite heterogeneous and require adhoc processing techniques. So far little emphasis has been given on developing an overall integrated environment for uniformly representing and querying different types of patterns. In recent research work [23] we have proposed such a framework: the *Pattern Warehouse*. The conceptual and logical model for the Pattern warehouse [24] defines an architecture of three modules: the *Pattern-Base*, where the pattern entities reside, the *Database* that is populated with raw data and the *Intermediate Mappings*, which is an indexing mechanism for tracing the connection between the patterns and the underlying data. Our late research interests focus on designing an efficient solution for the last module.

3.6 Context-Aware Databases

Information today is accessed and used in a global environment, where assumptions about data become less evident. Users with different backgrounds or viewpoints may interpret the same data in a different way. Moreover, the interpretation and suitability of data may depend on unpredictably changing conditions, like for example the current position of the user or the media he is using (laptop, mobile, PDA). To avoid such ambiguous situations the information provider needs to specify the context under which information becomes relevant. Conversely, information users can specify their own current context when requesting for data in order to denote the part that is relevant to their specific situation.

In our view the management of context should take place at the level of database systems in a uniform way, and context should be treated as a first-class citizen in data models and query languages. By incorporating context in semistructured data [25] and XML, we have shown that it is possible to have: a) management of data according to the interpretation frame, b) ability to pose *cross-world* queries that combine

information that hold under different contexts [26], c) personalization in a flexible and uniform manner and d) direct support for managing data and schema histories [27, 28].

Currently, we study the problem of incorporating context in the relational database model, which offers a strong formal background. We proposed the *ContextRelational* model (CR model) [29] that can be viewed as an extended relational model able to accommodate information entities that manifest different *facets* whose contents can vary in structure and value. Each facet of such a multi-faceted information entity is associated with a context, stating the conditions under which this facet holds. The operations of the CR model are context-aware extensions of the corresponding relational operations. We have developed a prototype implementation of the CR Model that demonstrates its basic principles.

4. WEB AND P2P DATA MANAGEMENT

We study models and techniques for managing Web data, and we explore alternative ways of interaction between databases and Web pages. Specifically, we work on querying methods for tree-structured data management based on partially specified queries. Also, we design effective architectures for P2P databases, and develop efficient query indexing and routing techniques for P2P spatial data. Finally, we design and implement new proxy architectures for dynamic Web pages.

4.1 Tree-structured Data Management

Huge volumes of Web data are organized in tree-structured form. Even if data is not stored natively in tree structures, export mechanisms make data publicly available in that form to enable its automatic processing by programs, scripts, and agents on the Web. XML data is by far the most prominent example. The recent proliferation of XML-based standards and technologies for managing data on the Web demonstrates the need for effective and efficient management of tree-structured data.

Querying tree-structured data sources usually requires resolving structural differences. Those differences appear because of the different possible ways of organizing the same data in tree-structures. Current tree-structured data query languages handle this issue in a procedural way. In this sense, the user should explicitly specify structural differences as part of the query itself. In fact, we identify a more general difficulty stemming from the fact that query formulation in current tree-structured data query approaches is strictly dependent on the structure of the data. Users cannot easily form queries without explicitly specifying the structural constraints.

Our ongoing research focuses on query languages that allow for the partial specification of the structure on which they are applied. Also, we study how syntactic and semantic information in tree-structured data can assist query evaluation.

In [30, 31, 32], we studied *dimension graphs*, which are semantically rich constructs that assist query formulation and evaluation on tree structured data. A key feature in our approach is that queries are not restricted by the structure of the trees. We applied our approach on querying effectively and efficiently multiple trees in the presence of structural differences and inconsistencies.

We further elaborated on the issue of *partially specified queries*, and we designed a full-fledged query language with *partially specified tree patterns* [33]. In this language, users can flexibly specify the structure fully, partially, or not at all in the queries. Since a query language needs to be complemented with query processing and optimization techniques, we have already started working on query containment and minimization issues for partially specified tree structured queries [34].

4.2 Peer-to-peer Database Systems

In the last few years, there has been a growing interest in the Peer-to-Peer (P2P) paradigm. In contrast to data integration architectures, P2P data sharing systems do not assume a mediated schema to which all sources of the system should conform. Sources store and manage their data locally, revealing only part of their schemas to the rest of the peers. Due to the lack of global schema, they express and answer queries based on their local schema. Peers also perform local coordination with their their one-hop neighbors, creating mappings semi-automatically.

In unstructured P2P systems, peers that join the network get their one-hop neighbors randomly, without taking into account that there might be available peers that better meet their need for information. Therefore, they have to direct queries not only to their neighbors, but to a greater part of the system. Query processing in unstructured P2P systems involves the propagation of the query on paths of bounded depth in the network. At each routing step, the query is rewritten to the schema of its new host based on the respective acquaintance mappings. A query may have to be rewritten several times from peer to peer till it reaches peers that are able to answer it sufficiently in terms of quality but also quantity. However, the successive rewritings decrease or restrict the information that can be returned by a query and, thus, they reduce the possibility of accurate query answering. It is the case that peers may not be able to sufficiently answer received queries not because their local schema does not match the initial query adequately, but because the incoming rewritten version is too reduced compared to the initial query.

To deal with the above problems, we have studied and implemented techniques to provide peers that share relational data in unstructured P2P networks with accurate answers to locally posed queries without any global schema information [35]. Our approach employs a learning feature to gradually cluster nodes with semantically similar local schemas. This is performed by utilizing only regular query traffic. Based on learning results, mappings between remote peers are gradually built on their specific common interests.

Moreover, we are interested in P2P systems hosting multi-dimensional data. Until recently, research has focused mostly on P2P systems that host one-dimensional data (i.e. strings, numbers, etc). However, the need for P2P applications with multi-dimensional data is emerging. Yet, most existing indexing and search techniques are not suitable for such applications. Most indices for multi-dimensional data have been developed for centralized environments, while, at the same time, existing distributed indices for P2P networks aim at one-dimensional data. Our focus is on structured P2P systems that share spatial information. We suggest [36], a totally decentralized indexing and routing technique that is suitable for spatial data. Our technique can be used in P2P applications in which spatial information of various sizes can be dynamically inserted or deleted, and peers can join or

leave. The proposed technique preserves well locality, and supports efficient routing especially for popular and close areas.

We also investigate P2P and mobile agent architectures based on active database technology [37, 38]. We argue that employing ECA rules both for answering queries and deploying agents leads to an efficient query processing technique. Furthermore, the proposed mobile agent system architecture offers a number of advantages due to the performance and scalability that can be achieved using Active Databases.

As a new research direction, we study P2P architectures to support loosely coupled database communities. We have designed a flexible wrapping mechanism, based on RDFS schemas, for data sources that employ diverse local schema information. We use schema operators for such wrapping [39]. The operators are applied on RDF schema graphs available for the community, and produce new, integrated ones. Such integration is based on set-like semantics and gives an intuitive way in wrapping data sources. We have also implemented a query processing technique that does not require the existence of mapping rules during the propagation of the query in the sources. Under this technique, we are also able to retrieve answers, even in the case a query does not exactly match the schema of a local data source. Our ideas are implemented in SDQNET, a platform that supports semantic query processing in loosely coupled data sources [40].

4.3 Web Caching

Proxy caching is a commonly accepted methodology used to reduce Internet traffic, decrease back-end related user delays and generally improve Web performance. Numerous Web caching approaches have been proposed concerning static Web pages. Given that the usage of dynamically generated Web pages increases continuously, there is need for Web caching algorithms for dynamic Web pages as well.

Front-end caching approaches implement caching outside the site infrastructure, e.g. a proxy or a cache that resides at the edge of a Content Delivery Network). However, front-end caching is inadequate when it comes to caching dynamic Web pages, due to their low degree of reusability and strong dependency on the back-end site infrastructure. This is because the request of a dynamic Web page depends on client-defined input parameters. Serving such a request using a cache could be possible only if a cached dynamic Web page had been produced from the same application with the same input parameters, a rather rare situation. Even in that case, the creation of dynamic Web pages depends on client-related information (e.g. through the use of cookies).

In order to overcome these obstacles, we have studied alternative front-end Web caching approaches. In particular, we have designed and implemented a new proxy architecture [41] for dynamic Web pages. In our approach, caching is performed on the generation process of the dynamic Web pages and not the pages themselves. Our current efforts focus on replacement policies based on group-oriented caching and pre-fetching algorithms that improve the performance of a Web caching system.

5. ACKNOWLEDGMENTS

The research achievements at KDBS Lab is the result of the work of many present and past members of the group. We would like to mention here all past and current PhD students who have greatly contributed into making KDBSL a great place to be: (past PhDs) D. Arkoumanis, T. Dalamagas, N. Karayannidis, M. Koubarakis, S. Ligoudistianos, A. Maniatis, D. Papadias, A. Simitsis, S. Skiadopoulos, Y. Stavrakas, E. Stefanakis, Y. Theodoridis, A. Tsois, P. Vassiliadis - (current PhD students) G. Adamopoulos, S. Athanasiou, P. Bouros, A. Dimitriou, K. Gavardinas, P. Georgantas, V. Kantere, Y. Kouvaras, G. Papastefanatos, K. Patroumpas, I. Roussos, D. Sacharidis, D. Skoutas, S. Souldatos, L. Stamatogiannakis, M. Terrovitis, M. Veliskakis, P. Xeros.

6. REFERENCES

[1] M. Jahangiri, D. Sacharidis, and C. Shahabi. Shift-Split: I/O Efficient Maintenance of Wavelet-transformed Multidimensional Data. In *Proceedings of the ACM SIGMOD'05 International Conference*, Baltimore, Maryland, Jun 14-16, 2005.

[2] G. Cormode, M. Garofalakis, and D. Sacharidis. Fast Approximate Wavelet Tracking on Streams. In *Proceedings of the EDBT'06 International Conference*, Munich, Germany, Mar 26-30, 2006.

[3] K. Patroumpas, T. Sellis. Window Specification over Data Streams. In *Proceedings of the ICSNW'06 International Conference*, Munich, Germany, Mar 30, 2006.

[4] K. Patroumpas, T. Sellis. Managing Trajectories of Moving Objects as Data Streams. In *Proceedings of the STDBM'04 Workshop*, Toronto, Canada, Aug 30, 2004.

[5] M. Potamias, K. Patroumpas, T. Sellis. Sampling Trajectory Streams with Spatiotemporal Criteria. In *Proceedings of the SSDBM'06 International Conference*, Vienna, Austria, Jul 3-5, 2006.

[6] A. Simitsis. Modeling and Optimization of Extraction-Transformation-Loading (ETL) Processes in Data Warehouse Environments. *PhD Thesis*, Athens, Greece, 2004.

[7] P. Vassiliadis, A. Simitsis, S. Skiadopoulos. Conceptual Modeling for ETL Processes. In *Proceedings of the DOLAP'02 International Workshop*, McLean, USA, Nov 8, 2002.

[8] P. Vassiliadis, A. Simitsis, S. Skiadopoulos. On the Logical Modeling of ETL Processes. In *Proceedings of the CAiSE'02 International Conference*, Toronto, Canada, May 27 - 31, 2002.

[9] P. Vassiliadis, A. Simitsis, P. Georgantas, M. Terrovitis. A Framework for the Design of ETL Scenarios. In *Proceedings of the CAiSE'03 International Conference*, Velden, Austria, Jun 16-20, 2003.

[10] A. Simitsis, P. Vassiliadis, M. Terrovitis, S. Skiadopoulos. Graph-Based Modeling of ETL Activities with Multi-Level Transformations and Updates. In *Proceedings of the DaWaK'05 International Conference* , Copenhagen, Denmark, Aug 22-26, 2005.

[11] P. Vassiliadis, A. Simitsis, S. Skiadopoulos. Modeling ETL Activities as Graphs. In *Proceedings of the DMDW'02 International Workdhop*, Toronto, Canada, May 27, 2002.

[12] P. Vassiliadis, A. Simitsis, M. Terrovitis, S. Skiadopoulos. Blueprints and Measures for ETL

Workflows. In *Proceedings of the ER'05 International Conference*, Klagenfurt, Austria, Oct 24-28, 2005.

[13] A. Simitsis. Mapping Conceptual to Logical Models for ETL Processes. In *Proceedings of the DOLAP'05 International Workshop*, Bremen, Germany, Nov 4-5, 2005.

[14] A. Simitsis, P. Vassiliadis, T. Sellis. Optimizing ETL Processes in Data Warehouse Environments. In *Proceedings of the ICDE'05 International Conference*, Tokyo, Japan, Apr 5-8, 2005.

[15] A. Simitsis, P. Vassiliadis, T. Sellis. State-Space Optimization of ETL Workflows. In *IEEE TKDE*, 17(10), 2005.

[16] P. Vassiliadis, A. Simitsis, P. Georgantas, M. Terrovitis, S. Skiadopoulos. A Generic and Customizable Framework for the Design of ETL Scenarios. In *Information Systems*, 30(7), 2005.

[17] G. Papastefanatos, P. Vassiliadis, Y. Vassiliou. Adaptive Query Formulation to Handle Database Evolution. In *Proceedings of the CAiSE'06 Forum*, Luxembourg, Grand-Duchy of Luxembourg, Jun 9, 2006.

[18] G. Papastefanatos, K. Kyzirakos, P. Vassiliadis, Y. Vassiliou. Hecataeus: A Framework for Representing SQL Constructs as Graphs. In *Proceedings of the EMMSAD'05 International Workshop*, Porto, Portugal, Jun 13-17, 2005.

[19] G. Koutrika, A. Simitsis, Y. Ioannidis. Précis: The Essence of a Query Answer. In *Proceedings of the ICDE'06 International Conference*, Atlanta, USA, Apr 3-7, 2006.

[20] A. Simitsis, G. Koutrika. Pattern-Based Query Answering. In *Proceedings of the PaRMa'06 International Workshop*, Munich, Germany, Mar 30, 2006.

[21] A. Simitsis, G. Koutrika. Comprehensible Answers to Précis Queries. In *Proceedings of the CAiSE'06 International Conference*, Luxembourg, Grand-Duchy of Luxembourg, Jun 5-9, 2006.

[22] M. Terrovitis, S. Passas, P. Vassiliadis, T. Sellis. A combination of trie-trees and inverted files for the indexing of set-valued attributes, 2006. *Submitted for publication.*

[23] S. Rizzi, E. Bertino, B. Catania, M. Golfarelli, M. Halkidi, M. Terrovitis, P. Vassiliadis, M. Vazirgiannis, E. Vrachnos. Towards a logical model for patterns. In *Proceedings of the ER'03 International Conference*, Chicago, USA, Oct 13-16, 2003.

[24] M. Terrovitis, P. Vassiliadis, S. Skiadopoulos, E. Bertino, B. Catania, A. Maddalena. Modeling and Language Support for the Management of Pattern-Bases. In *Proceedings of the SSDMB'04 International Conference*, Santorini Island, Greece, Jun 21-23, 2004.

[25] Y. Stavrakas, M. Gergatsoulis. Multidimensional Semistructured Data: Representing Context-Dependent Information on the Web. In *Proceedings of the CAiSE'02 International Conference*, Toronto, Canada, May 27-31, 2002.

[26] Y. Stavrakas, K., A. Efandis, T. Sellis. Implementing a Query Language for Context-dependent Semistructured Data. In *Proceedings of ADBIS'04 Conference*, Budapest, Hungary, Sep 22-25, 2004.

[27] Y. Stavrakas, M. Gergatsoulis, C. Doulkeridis, V. Zafeiris. Representing and Querying Histories of Semistructured Databases Using Multidimensional OEM. In *Information Systems*, 29(6), 2004.

[28] M. Gergatsoulis, Y. Stavrakas. Representing Changes in XML Documents Using Dimensions. In *Proceedings of the XSym'03 International Symposium*, Berlin, Germany, Sep 8, 2003.

[29] Y. Roussos, Y. Stavrakas, V. Pavlaki. Towards a Context-Aware Relational Model. In *Proceedings of the CRR'05 Workshop*, Paris, France, Jul 5-8, 2005.

[30] T. Dalamagas, D. Theodoratos, A. Koufopoulos, and V. Oria. Evaluation of Queries on Tree-Structured Data using Dimension Graphs. In *Proceedings of the IDEAS'05 Symposium*, Montreal, Canada, Jul 25-27, 2005.

[31] T. Dalamagas, D. Theodoratos, A.Koufopoulos, and I-Ting Liu. Semantic Integration of Tree-structured Data Using Dimension Graphs. In *Journal on Data Semantics*, IV, 2005.

[32] D. Theodoratos, and T. Dalamagas. Querying Tree-Structured Data using Dimension Graphs. In *Proceedings of the CAiSE'05 International Conference*, Porto, Portugal, Jun 13-17, 2005.

[33] D. Theodoratos, T. Dalamagas, A. Koufopoulos, and N. Gehani. Semantic Querying of Tree-Structured Data Sources Using Partially Specified Tree Patterns. In *Proceedings of the CIKM'05 Conference*, Bremen, Germany, Oct 31 - Nov 5, 2005.

[34] D. Theodoratos, T. Dalamagas, P. Placek, S. Souldatos and T. Sellis. Containment of Partially Specified Tree-Pattern Queries. In *Proceedings of the SSDBM'06 International Conference*, Vienna, Austria, Jul 3-5, 2006.

[35] V. Kantere, D. Tsoumakos, T. Sellis, N. Roussopoulos. GrouPeer: Dynamic Clustering of P2P Databases, 2006. *Submitted for publication.*

[36] V. Kantere, T. Sellis. Handling Spatial Data in P2P Systems, 2006. *Submitted for publication.*

[37] V. Kantere, A. Tsois. Using ECA Rules to Implement Mobile Query Agents for Fast-Evolving P2P Networks. In *Proceedings of the AAMAS'04 International Conference*, New York, USA, Aug 19-23, 2004.

[38] V. Kantere, I. Kiringa, J. Mylopoulos, A. Kementsientidis, M. Arenas. Coordinating P2P Databases Using ECA Rules. In *Proceedings of the DBISP2P'03 International Workshop*, Berlin, Germany, Sep 7-8, 2003.

[39] Z. Kaoudi, T. Dalamagas, T. Sellis. RDFSculpt: Managing RDF Schemas under Set-like Semantics. In *Proceedings of the ESWC'05 Conference*, Heraklion, Greece, May 29 - Jun 1, 2005.

[40] I. Spyropoulou, T. Dalamagas, T. Sellis. SDQNET: Semantic Distributed Querying in Loosely Coupled Data Sources, 2006. *Submitted for publication.*

[41] M. Veliskakis, Y. Roussos, P. Georgantas, T. Sellis. DOMProxy: Enabling Dynamic-Content Front-end Web Caching. In *Proceedings of the WCW'05 International Workshop*, Sofia Antipolis, France, Sep 12-13, 2005.

Consistent Query Answering in Databases*

Leopoldo Bertossi
Carleton University
School of Computer Science
Ottawa, Canada.
bertossi@scs.carleton.ca

1 Introduction

For several reasons databases may become inconsistent with respect to a given set of integrity constraints (ICs): (a) The DBMS have no mechanism to maintain certain classes of ICs. (b) New constraints are imposed on pre-existing, legacy data. (c) The ICs are soft, user, or informational constraints that are considered at query time, but without being necessarily enforced. (d) Data from different and autonomous sources are being integrated, in particular in mediator-based approaches.

In many cases cleaning the database from inconsistencies may not be an option, e.g. in virtual data integration, or doing it may be costly, non-deterministic, and may lead to loss of potentially useful data. Furthermore, it is quite likely that most of the data in the database is still "consistent". In consequence, an alternative approach to data cleaning consists in basically living with the inconsistent data, but making sure that the consistent data can be identified if desired, for example when queries are answered from the database.

Of course, if this approach is followed, the first problem that has to be confronted is the one characterizing in precise terms the notion of consistent data in a possibly inconsistent database. Such a definition was proposed in [1]. The basic intuition is that a piece of data in the database D is consistent if it is invariant under minimal forms of restoring the consistency of the database, i.e. it remains in every database instance D' that shares the schema with D, is consistent wrt the given ICs, and "minimally differs" from D. This natural intuition still depends on what we consider to be maximally close to the original instance. Several alternatives offer themselves, and we come back to this point in Section 4, but for the moment, in order to fix ideas, we stick to the notion of distance used in [1]: The set of database tuples either inserted or deleted into/from the database to restore consistency has to be made minimal under set inclusion. These repairs will be called S-repairs.

We will concentrate on relational databases. Thus, we start from a fixed relational schema $\mathcal{S} = (\mathcal{U}, \mathcal{R}, \mathcal{B})$, where $\mathcal{U}$ is the possibly infinite database domain, $\mathcal{R}$ is a set of database predicates, and $\mathcal{B}$ is a set of built-in predicates, e.g. comparison predicates. This schema determines a language $\mathcal{L}(\mathcal{S})$ of first-order predicate logic. A database instance D compatible with $\mathcal{S}$ can be seen as a finite collection of ground atoms of the form $R(c_1, ..., c_n)$ that we will call *database tuples*, where R is a predicate in $\mathcal{R}$ and $c_1, ..., c_n$ are constants in $\mathcal{U}$. Built-in predicates have a fixed extension in every database instance, not subject to changes. Integrity constraints are sentences in $\mathcal{L}(\mathcal{S})$.

A database instance D is consistent wrt a finite set IC of integrity constraints if D satisfies IC as a first-order structure, denoted $D \models IC$. Otherwise, D is inconsistent wrt IC. A *repair* D' of D is another instance over schema $\mathcal{S}$ that satisfies IC and makes $\Delta(D, D') = (D \setminus D') \cup (D' \setminus D)$ minimal under set inclusion. We denote with $Rep(D, IC)$ the set of repairs of D wrt IC.

Example 1 Consider a schema with relation $P(A, B, C)$ and the functional dependency (FD) $A \rightarrow B$. The inconsistent instance $D = \{P(a, b, c),\ P(a, c, d),\ P(a, c, e),\ P(b, f, g)\}$ has two repairs: $D_1 = \{P(a, b, c), P(b, f, g)\}$ and $D_2 = \{P(a, c, d),\ P(a, c, e), P(b, f, g)\}$, because the symmetric set differences with the original instance, $\Delta(D, D_1), \Delta(D, D_2)$, are minimal under set inclusion. □

We are not particularly interested in the repairs of the database, or better, in computing them, but in the consistent data in the database, more specifically in consistent answers to queries. From this point of view, we use the repairs as auxiliary objects.[1]

A query is a first-order formula $Q(x_1, \ldots, x_n)$ in $\mathcal{L}(\mathcal{S})$, with free variables $x_1, \ldots, x_n$, $n \geq 0$, and a tuple $\bar{t} = (t_1, \ldots, t_n) \in \mathcal{U}^n$ is a *consistent answer* to Q in D wrt IC if for every $D' \in Rep(D, IC) : D' \models Q[\bar{t}]$, i.e. Q becomes true in D' when the variables $x_1, \ldots, x_n$ take the values $t_1, \ldots, t_n$, resp. When $n = 0$, i.e. the query is boolean, yes is the consistent answer if $D' \models Q$ for every $D' \in Rep(D, IC)$. Otherwise, no is the consistent answer.

Example 2 (example 1 continued) The query $Q_1(x) : \exists y \exists z P(x, y, x)$ has $(a), (b)$ as the only consistent answers, because they are the only usual answers obtained from the two repairs. The query $Q_2(y, z) : \exists x P(x, y, z)$

---

* **Database Principles Column.**

Editor: Leonid Libkin, Department of Computer Science, University of Toronto, Toronto, Canada M5S 3H5. E-mail: libkin@cs.toronto.edu.

[1] However, in some applications computing repairs may be more relevant that computing consistent answers to queries, cf. Section 4.1.

has (f, g) as consistent answer. Finally, $Q_3 : P(x, y, x)$ has (b, f, g) as its consistent answer. □

Since these definitions were introduced in [1], several papers have been written on the subject of *consistent query answering* (CQA). For an earlier survey, more references, and related work, see [12].

2 Computing Consistent Answers

As emphasized above, in most applications, computing consistent answers to queries is the natural problem to solve. In an ideal situation, those answers should be obtained by querying the given, inconsistent databases, hopefully avoiding as much as possible the explicit computation of the repairs and checking candidate answers in them. It is easy to see that there may be exponentially many repairs in the size of the database.

The first algorithm for computing consistent answers to queries in this spirit was proposed in [1]. It is a query rewriting algorithm, i.e. the original query Q, expecting for consistent answers from D, is rewritten into a new query Q' in such a way that the usual answers to Q' in D are the consistent answers to Q from D. Query Q' can be computed iteratively by appending to Q additional conditions, the *residues*, that are obtained from the interaction between Q and the ICs, in order to locally enforce the satisfaction of the ICs.

Example 3 (example 2 continued) Query Q_3 can be rewritten into the query
$Q'(x, y, z)$: $P(x, y, z) \wedge \forall v \forall w(\neg P(x, v, w) \vee y = v)$, asking for those tuples (x, y, z) in P for which x does not have and associated u different from y. The *residue* can be obtained by resolution between the original query and the clausal form $\forall uvv'ww'(\neg P(u, v, w) \vee \neg P(u, v', w') \vee v = v')$ of the FD. □

In this example the original query was rewritten into a new first-order query, that could be expressed in SQL, as the original query, and posed to instance D. The usual answers to the rewritten query are exactly the consistent answers to the original query.

Notice that there may be several ICs and also several literals in the query that "logically interact" with the ICs, so that this residue appending process has to be iterated. In [1] conditions for soundness, completeness, and finite termination are identified. This algorithm is guaranteed to produce a first-order rewriting for projection-free conjunctive queries and universal ICs, i.e. that are logically equivalent to a quantifier-free formula preceded by a block of universal quantifiers.

A system implementation for computing consistent answers for this class of queries and ICs that is based on this methodology is reported in [21]. Another system for the same class of queries, but for denial ICs, i.e. ICs that logically equivalent to one of the form

$$\forall \bar{x} \neg (A_1 \wedge \ldots \wedge A_m \wedge \gamma), \qquad (1)$$

where each A_i is a database atom and γ is a conjunction of comparison atoms, is described in [22], however the latter is based on a graph-theoretic representation of repairs and not on syntactic query rewriting (cf. Section 3).

Given the -as we will see- intrinsic limitations of methodology to obtain consistent answers based on first-order query rewriting, a more general approach based on specifying database repairs as the the models of a logic program was proposed [4, 35]. More specifically, it is possible to use a disjunctive logic program with stable model semantics [34] and annotation constants, whose rules capture the violations of the ICs in the bodies and propose alternative ways of locally restoring consistency by means of their disjunctive heads [5].

Example 4 (example 1 continued) The repair program[2] $\Pi(D, FD)$ contains the original database atoms as facts, and the rule $P'(x, y, z, \mathbf{d}) \vee P'(x, u, v, \mathbf{d}) \leftarrow P(x, y, z), P(x, u, v), y \neq u$, where the extra annotation $\mathbf{d}$ used in the head indicates that one of the conflicting tuples should be deleted. The body is satisfied when the FD is violated.

The annotation constant $\mathbf{r}$ is used to read off the database tuples that are inside a repair $P(\bar{x}, \mathbf{r}) \leftarrow P(\bar{x}), \ not\ P'(\bar{x}, \mathbf{d})$, i.e. they contain those database atoms that were not deleted from the original database.

The stable models of the repair program are in one-to-one correspondence with the database repairs; and if the consistent answers to the query $\exists y \exists z P(x, y, z)$ are to be computed, the repair program can be extended with the query rule $Ans(x) \leftarrow P'(x, y, z, \mathbf{r})$. The consistent answers will be those in the extension of the Ans predicate when the extended program is evaluated according to the skeptical (or cautious or certain) stable model semantics, i.e. the one that sanctions as true whatever is true of all stable models. □

This methodology is provably correct and complete for any combination of first-order queries (or in extensions of Datalog), universal ICs, and referential ICs; the latter containing no referential cycles [6]. In spite of its generality, this approach may be too expensive given that query evaluation of disjunctive logic programs under the skeptical stable model semantics is Π_2^P-complete [24]. However, as we will see in Section 3, in some cases we need the expressive power of this kind of programs. Optimizations of these repairs programs have been presented in [27, 6, 20].

3 Complexity of Consistent Query Answering

It is clear that for those cases where the first-order query rewriting methodology for CQA works, the complexity of computing consistent answers is polynomial in the size

[2] In a very simplified form wrt the general methodology, to show the gist of the approach.

$|D|$ of the underlying database D, i.e. it is in $PTIME$ in *data complexity*, a usual measure of complexity in databases.

The first lower-bounds for CQA were obtained for sets FD of functional dependencies and scalar (group-by free) aggregate queries [2]. Since in this case the query returns a single numerical value from a database, which can be different in every repair, the semantics for CQA used in this context is the *range semantics*. In this case the consistent answer to an aggregate query Q is an optimal numerical interval $I_{op} = (op_L, op_U)$, such that for every $D' \in Rep(D, IC)$, the numerical answer $Q(D')$ to Q in D' falls in I_{op}.

Example 5 Consider schema $P(A, B)$, the FD $A \rightarrow B$, and the inconsistent instance $D = \{P(a, 4), P(a, 9), P(b, 3), P(c, 5)\}$. The repairs are $D_1 = \{P(a, 4), P(b, 3), P(c, 5)\}$ and $D_2 = \{P(a, 9), P(b, 3), P(c, 5)\}$. Here, the consistent answer to the query $min(B)$ should be 3, the minimum in both repairs, whereas for the query $sum(B)$, we get answers 13 and 17 from D_1, D_2, resp. Under the range semantics for CQA, the answer is the optimal interval $[13, 17]$, the most informative interval where the answers to the original query from the different repairs will be found. □

Finding the optimal lower and upper bounds op_L, op_U, resp., become maximization and minimization problems, resp., that have associated decision problems: $CQA_L(Q, FD) = \{(D, k) \mid k \leq op_L\}$, and $CQA_U(Q, FD) = \{(D, k) \mid op_U \leq k\}$, resp. As usual, we are interested in the data complexity of these problems.

In order to obtain complexity bounds and algorithms for these decision problems, it is useful to provide a graph-theoretic characterization of repairs, which are in one-to-one correspondence with the (set-theoretically) maximal independent sets of the *conflict graph* $\mathcal{G}(D, FD)$ associated to instance D and FDs FD. The vertices of $\mathcal{G}(D, FD)$ are the database tuples in D, and any two vertices are connected by an edge if they simultaneously participate in the violation of one of the FDs in FD.

A classification of the complexity of CQA for the main aggregate queries is given in [3]. Basically already with two FDs the decision problems associated to CQA become NP-complete.

For first-order queries $Q(\bar{x})$ the decision problem of CQA is $CQA(Q, IC) = \{(D, \bar{t}) \mid \bar{t}$ is a consistent answer to Q from D wrt $IC\}$.

The graph-theoretic methods introduced in [2] were extended to deal with the CQA problem for sets IC of denial constraints, that include FDs [23]. In this case, instead of conflict graphs we find *conflict hyper-graphs*, whose vertices are the database tuples, but now hyper-edges connect all ad only the tuples that simultaneously violate a denial of the form (1). For denial constraints repairs can be obtained via tuple deletions alone, and there is still a one-to-one correspondence between the S-repairs and the maximal independent sets in the hyper-graph.

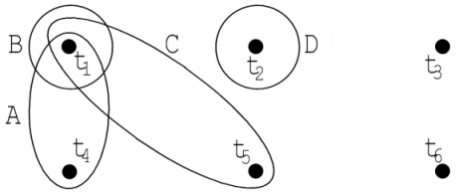
Example 6 The schema has tables $Client(ID, A, C)$, with attributes ID, an identification key, age and credit line; and $Buy(ID, I, P)$, with key $\{ID, I\}$, for clients buying items (I) at certain prices (P). There are denial constraints $IC_1 : \forall ID, P, A, C \neg($ $Buy(ID, I, P), Client(ID, A, C), A < 18, P > 25)$ and $IC_2 : \forall ID, A, C \neg($ $Client($ $ID, A, C), A < 18, C > 50)$, requiring that people younger than 18 cannot spend more than 25 on one item nor have a credit line higher than 50 in the store. The following is an inconsistent instance D (we use an extra column to denote the tuple)

Client

ID	A	C	
1	15	52	t_1
2	16	51	t_2
3	60	900	t_3

Buy

ID	I	P	
1	CD	27	t_4
1	DVD	26	t_5
3	DVD	40	t_6



The hyper-edges are $\{t_1, t_4\}$ and $\{t_1, t_5\}$ for IC_1 and $\{t_1\}$ and $\{t_2\}$ for IC_2. The figure shows the hyper-graph associated to D and the denial constraints (the key constraints are satisfied in this example). For example, the instance consisting of database tuples $\{t_3, t_4, t_5, t_6\}$ would be an S-repair, that corresponds to maximal independent set. □

In [23] the complexity analysis was extended to denials in combination with inclusion dependencies of the form $\forall \bar{x} \exists \bar{y}(P(\bar{x}) \rightarrow R(\bar{x}', \bar{y}))$, with $\bar{x}' \subseteq \bar{x}$. These constraints can be repaired by deleting the inconsistent tuple in P or inserting a tuple into R. However [23] considers only repairs that are obtained through tuple deletions. This class of dependencies include universal inclusion dependencies and referential ICs. In [23] it shown that for conjunctive queries with projections and FDs, the problem of CQA becomes $coNP$-complete; and with inclusion dependencies, even Π_2^P-complete.

Similar complexity results were obtained in [18] under a repair semantics that allows for both tuple deletions and insertions when FDs and inclusion dependencies are combined. In this case, CQA becomes undecidable due to the possible presence of cycles among the inclusion dependencies and the possibility of using arbitrary elements from the underlying infinite database domain $\mathcal{U}$ when tuples are inserted.

Interestingly, the tight complexity upper bound for decidable CQA coincides with the complexity upper bound

of query evaluation from the repair programs that specify the repairs, which shows that in the worst cases, the expressive power of disjunctive logic programs is necessary.

The complexity results mentioned above show that first-order query rewriting for CQA has intrinsic limitations, which makes it necessary to rewrite the original query into a logic program as opposed to a first-order query. As shown in [6], this program may be simpler than disjunctive. There, classes of ICs are identified for which the disjunctive programs can be transformed into a (non-disjunctive) normal programs, whose query evaluation complexity becomes $coNP$-complete [24].

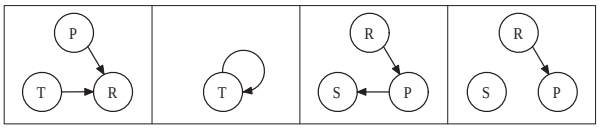
From the previous discussion, we can see that for FDs and conjunctive queries we can go from $PTIME$, e.g. for projection-free conjunctive queries, to $coNP$-completeness for some particular classes of existentially quantified conjunctive queries. In consequence, a natural problem was to identify classes of conjunctive queries with projection for which CQA is still tractable. The first such classes were identified in [23], and polynomial time algorithms were developed appealing to the conflict graph-based representation of repairs. This analysis was extended in detail in [32], were a tight class, $\mathcal{C}_{Tree}$, of conjunctive queries was identified for which CQA is still tractable.

More precisely, as defined in [32], the *join graph* $\mathcal{G}(Q)$ of a boolean conjunctive query Q is a directed graph, whose vertices are the database atoms in Q. There is an arc from L to L' if $L \neq L'$ and there is a variable w that occurs at the position of a non-key attribute in L and also occurs in L'. Furthermore, there is a self-loop at L if there is a variable that occurs at the position of a non-key attribute in L, and at least twice in L. By definition, a query Q belongs to the class $\mathcal{C}_{Tree}$ if Q does not have repeated relations symbols and $\mathcal{G}(Q)$ is a forest and every non-key to key join of Q is full i.e. involves the whole key. CQA becomes tractable for queries in $\mathcal{C}_{Tree}$.[3]

Algorithms for consistently answering queries in the class $\mathcal{C}_{Tree}$ wrt FDs were implemented and reported in [33]. Tractable classes of union of queries in $\mathcal{C}_{Tree}$ are identified in [37].

Example 7 Consider the following boolean queries, where the primary keys of the relations involved are underlined and all the variables are existentially quantified: $Q_1 : P(\underline{x}, y) \wedge R(\underline{y}, w) \wedge T(\underline{u}, \underline{v}, y)$; $Q_2 : T(\underline{x}, \underline{y}, y)$; $Q_3 : R(\underline{x}, y) \wedge P(\underline{y}, z) \wedge S(\underline{z}, \underline{u})$; and $Q_4 : R(\underline{x}, y) \wedge S(\underline{w}, z) \wedge P(\underline{y}, u)$. The four associated join graphs, respectively, are shown below. $\mathcal{G}(Q_1)$ and $\mathcal{G}(Q_2)$ are not forests, therefore their queries are not in $\mathcal{C}_{Tree}$. Even though $\mathcal{G}(Q_3)$ is a forest, since it has a non-key to key join that is not full, it does not belong to $\mathcal{C}_{Tree}$.

[3]Open conjunctive queries can be accommodated in this class by treating the free variables in them as constants in the definition of their join graph.



Q_4 is in $\mathcal{C}_{Tree}$ because $\mathcal{G}(Q_3)$ is a forest and all the non-key to key joins are full. □

4 Other Repair Semantics

We have seen that the notion of consistent answer is defined in terms of the auxiliary notion of repair, which includes in its turn a notion of minimal restoration of consistency. Most of the research in CQA has been concentrated on *repairs* that differ from the original database by a *minimal set of whole (inserted or deleted) tuples under set inclusion*, as were defined above.

Depending on the application domain, or complexity issues, other notions of repair could be considered and studied, in particular, in terms of their impact on CQA.

4.1 Attribute-based repairs

In Example 6, if we consider the repair semantics introduced before, inconsistencies would be solved by deleting complete database tuples from the database. In [48, 31] a different kind of repairs was introduced; they are obtained by changing some attribute values in existing tuples. We call them, attribute-based repairs (simply A-repairs). Except for [48], all the A-repairs considered so far [31, 10, 29] minimize a numerical aggregation function over the attribute-value changes.

More precisely, we can use a numerical function w defined on tuples of the form $(R(\bar{t}), A, newValue)$, where R indicates a tuple in the original instance, A the attribute of R, and $newValue$ is the new value that attribute A gets in $R(\bar{t})$. Then, we minimize an aggregation function g over the values taken by w in the instance, that is the A-repairs are consistent instances that minimize the value of g. Typically [31, 29], $w(R(\bar{t}), A, newValue) = \delta(R(\bar{t}).A, newValue)$, i.e. it gives 1 if there is a change, and 0 otherwise. Next g is the sum, that is, the number of changes is minimized.

Most of the research around this kind of repairs has been motivated by the real need to repair the original database without getting rid of the whole conflicting database tuples. Typical applications are related to census data [31, 10], where census forms have to be *edited* in order to make them consistent wrt to certain denial constraints that forbid certain combinations of data. For example, in household data associated to a census, we may consider that there is a mistake if a child under 5 shows "married" as marital status.

In this kind of statistical applications, a main focus of interest is on obtaining concrete repairs. Consistent query answering based on A-repairs is a natural problem for aggregate numerical queries under the range semantics (cf. Section 3).

In [10], the class of attribute repairs was defined in terms of the functions square difference and sum (w and g above, resp.), i.e. repairs minimize the sum of square differences of numerical values in fixable numerical attributes taking integer values. The basic assumptions are that these numerical values are associated to -non necessarily numerical- values taken by a key, and that the key constraint is satisfied by the original database and its repairs (which makes it possible to compare numerical values associated to values of the key). For example, in a census database, this assumption means that a a household code is unique, but values associated to this code may be incorrect wrt to a separate set of denial constraints, e.g. number of people in the house; or the age of the oldest son, etc.

Numerical distances capture in a better way the numerical nature of some relevant attributes, and also the fact that when we repair (in this case, correct mistakes), the corrections are minimal in numerical difference while satisfying the constraints.

Example 8 (example 6 continued) The A-repairs of D that minimize the square distance are

D':

Client

ID	A	C	
1	15	50	t_1'
2	16	50	t_2'
3	60	900	t_3

Buy

ID	I	P	
1	CD	25	t_4'
1	DVD	25	t_5'
3	DVD	40	t_6

D'':

Client

ID	A	C	
1	18	52	t_1''
2	16	50	t_2''
3	60	900	t_3

Buy

ID	I	P	
1	CD	27	t_4
1	DVD	26	t_5
3	DVD	40	t_6

These repairs have the square distances to the original instance $1^2 + 2^2 + 1^2 + 2^2 = 10$ and $1^2 + 3^2 = 10$, resp.; and they satisfy the denial constraints. □

A relevant decision problem to solve in this context consists in determining if there is a repair at a distance smaller than a given budget to the initial instance. This problem turns out to be NP-hard [10] even for the natural class of denial constraints that are "local", as those in Example 6 (intuitively, inconsistencies can be solved by concentrating on changes that are local to each denial). Even more, for this class of constraints there is no polynomial-time approximation schema (PTAS) for the associated optimization problem. However, it is possible to provide a polynomial-time approximation algorithm that gives an answers within a constant factor of the optimal solution.

Another relevant decision problem is CQA for aggregate queries under the range semantics. This problem is $coNP$-hard for the most common scalar aggregate functions, and even for one database atom denial constraints (like IC_2 in Example 6). For example, for sum no PTAS exists, but, again, a polynomial-time and constant factor approximation algorithm can be given for the optimal bounds of the range semantics [10].

Attribute-based repairs under aggregation constraints have been investigated in [29].

4.2 Cardinality-based repairs

Less attention than CQA based on S-repairs (introduced in Section 1) has received the same problem relative to "cardinality-based repairs" of the original database (simply C-repairs) that minimize *the number* of whole database tuples by which the instances differ.

Example 9 (example 1 continued) Among the S-repairs D_1 and D_2, only D_2 is a cardinality-based repair, because the cardinality $|\Delta(D, D_2)|$ of the symmetric set difference becomes a minimum. □

The complexity of CQA wrt denial constraints under the C-repair semantics has been investigated in [44]. Similarly as with S-repairs, the C-repairs are in a one-to-one correspondence with the *maximum* (in cardinality) independent sets (MISs) in the conflict hyper-graph determined by the original instance and the constraints.

A database tuple is consistently true if it belong to every MIS. In contrast to the S-repair semantics, a database tuple in the original instance may not belong to any MIS. Actually we obtain that CQA for ground atomic queries and denial constraints under the C-repair semantics is $P^{NP(log\ n)}$-complete [44], which contrast with the polynomial time complexity for the same problem wrt the S-semantics [23]. That is, for C-repairs we need polynomial-time algorithm that makes $O(log(n))$ calls to an NP-oracle, where n is the size of the original database.

4.3 CQA under null values

The database repairs considered so far have not explicitly taken into account that a database may have null values and that consistency can also be restored using them, e.g. for referential integrity constraints. There are many different semantics for null values and for databases that represent incomplete information. Some of them consider different kinds of null values, or different but labelled nulls of the same kind, etc. In commercial DBMS we find only one null value. There is something like a semantics for its use in the SQL standard, which is not always implemented in real systems.

In [15] this issue was systematically studied, considering databases with possibly many different occurrences of the same $null$ constant, as found in database practice. Also repairs of referential ICs made use of this constant, as an alternative to the co-existing possibility of deleting the violating tuple. This makes it necessary to provide a precise and uniform semantics for IC satisfaction in the presence of null values that extends and is compatible with the way commercial systems deal with ICs. This semantics was introduced in [15], together with a new repair semantics that privileges the use null values over arbitrary constants in the domain when tuples are inserted to satisfy existential ICs.

Example 10 Consider the ICs $\forall xy(P(x,y) \rightarrow T(x))$, $\forall x(T(x) \rightarrow \exists y P(y,x))$, and the inconsistent database $D = \{P(a,b), P(null,a), T(c)\}$. In this case, we have a referential cycle in the set of ICs. According to the modified repair semantics, the four repairs are:

i	D_i
1	$\{P(a,b), P(null,a), T(c), P(null,c), T(a)\}$
2	$\{P(a,b), P(null,a), T(a)\}$
3	$\{P(null,a), T(c), P(null,c)\}$
4	$\{P(null,a)\}$

We obtain a finite number of repairs (each with finite extension). If we repaired the database by using the non-null constants in the infinite domain with the S-repair semantics of [1], we would obtain an infinite number of repairs and infinitely many of them with infinite extension, as considered in [18]. □

In contrast to the undecidability result for CQA under classic S-repair semantics for referential ICs reported in [18], with the null-value based semantics decidability is reached.

5 Dynamic Aspects of CQA

It is in the context of the C-repair semantics (cf. Section 4.2) that dynamic aspects of CQA have been investigated for the first time. More precisely, in [44] the *incremental complexity* of CQA is considered. In this case, the original database D, of size n, is consistent wrt a set of ICs, but after a sequence $U : u_1, \ldots, u_m$ of basic updates u_i that are insertions on one tuple, deletion of one tuple, or a change of an attribute value in a tuple, the updated database $U(D)$ may become inconsistent. Incremental complexity of CQA is the complexity of CQA wrt the updated instance $U(D)$.

For the C-repair semantics, first-order boolean queries, and denial constraints, incremental CQA is in *PTIME* in the size of D, however a naive algorithm that gives this upper bound, namely $O(n^{p(m)})$ with p a polynomial, gives an exponential complexity in m, the size of the update sequence [44].

It is possible to prove that this problem, still under the C-repair semantics, for functional dependencies and ground atomic queries, is *fixed parameter tractable* (FPT) [38, 30, 25], i.e. there is an algorithm that of order $O(f(m) \cdot n^c)$, with f a function of m alone, and c a constant. The parameter here is m, the size of the update sequence. The algorithm given in [44] appeals to the fixed parameter tractability of vertex cover. Using the membership to FPT of the d-hitting set problem (finding the size of a minimum hitting set for a hyper-graph with hyper-edges bounded in size by d) [46], it is possible to prove that incremental CQA under the C-repair semantics also belongs to FPT for denial constraints that have a fixed number d of database tuples, which is a natural assumption [44].

Summarizing, for boolean queries and denial ICs, incremental CQA is $PTIME$ on n under the C-repair semantics. However, under the same conditions but now the S-repair semantics, incremental CQA is $coNP$-complete on n [44]. The difference with the C-repair semantics becomes more clear if we consider that the cost of a C-repair cannot exceed the size m of the update, whereas the cost of an S-repair may be unbounded wrt the same size.

Example 11 Consider the schema $R(\cdot), S(\cdot)$, with denial constraint $\forall x \forall y \neg(R(x) \wedge S(y))$. The following is a consistent instance $D = \{R(1), \ldots, R(n)\}$, with empty table for S. After the update $U : insert(S(0))$, the database becomes inconsistent, and the S-repairs are $\{R(1), \ldots, R(n)\}$ and $\{S(0)\}$. Only the former is a C-repair, at a distance 1 from D, but the second S-repair is at a distance n. □

Finally, we should mention that incremental CQA wrt denial constraints and atomic queries under the A-repair semantics (cf. Section 4.1) becomes P^{NP}-hard on n [44]. In this case, after the update sequence (consisting of insertions, deletions and one attribute value changes), the database may become inconsistent, but it can only be repaired through attribute value changes.

It would be interesting to analyze the complexity of incremental CQA from the point of view of *dynamic complexity* [40, 47] and *incremental complexity* as introduced in [45]. Here the database is not necessarily consistent before the update, but auxiliary data structures can be used for incremental computation of CQA.

6 Data Exchange and Integration

Consistent query answering is related to several other areas, most prominently to virtual data integration, data exchange, and peer-to-peer data exchange.

As emphasized in [11], in data exchange, where data is shipped from a source database in order to populate a target schema, integrity constraints imposed at the target level have to be kept satisfied [41]. Instead of restoring the consistency of data at the target *a posteriori*, after the population process, a more appealing alternative would take the ICs at the target into account when the data mappings between the source and the target are being established and/or used.

In virtual data integration [43] there is no centralized consistency maintenance mechanism that makes the data in the mediated system satisfy certain global ICs. Again, these ICs have to be captured by the mappings between the sources and the global schema or at query time [8], when global queries are being answered. Both in data exchange and virtual data integration, the plans for data transfer or query answering have to deal with potential inconsistencies of data.

Mediator-based data integration is a natural scenario for applying CQA: We expect to retrieve only the information from the global, virtual database that is consistent wrt to a given set of global ICs. In this scenario, some new issues appear, like characterizing repairs of global virtual instances, and retrieve consistent answers at query time, considering that the data is in the sources.

In the following we will illustrate the approach developed in [7, 13, 14] by means of an extended example. For a general overview, detailed discussion of related work, and additional results cf. [8].

We will assume that the *local-as-view* (LAV)[4] approach is adopted, which means that the relations at the source schemas are described as views over the global schema that the mediator offers to the users. The underlying domain is $\mathcal{U} = \{a, b, c, \dots\}$, the global schema is Global system $\mathcal{G}$ consists of relations $P(A, B)$ and $R(C, D)$, and the source relations are S_1, S_2, with extensions $s_1 = \{S(a, b)\}, s_2 = \{S_2(a, c), S_2(d, e)\}$. The source relations are defined by (using Datalog notation):

$$\begin{aligned} S_1(X, Z) &\leftarrow P(X, Y),\ R(Y, Z). \\ S_2(X, Y) &\leftarrow P(X, Y). \end{aligned}$$

The global relations P, R can be materialized in different ways, still satisfying the view definitions (or source descriptions); so different global instances are possible become possible. If the sources are declared as *open* (or incomplete), a global (material) instance D is *legal* if the view definitions applied to it compute extensions $S_1(D), S_2(D)$ such that $s_1 \subseteq S_1(D)$ and $s_2 \subseteq S_2(D)$. That is, each source relation contains a possibly proper subset of the data of its kind in the global system.

The *certain answers* to a global query Q, i.e. expressed in terms of relations P and R, are those that can be (classically) obtained from every possible legal instance. Among the legal instances we can distinguish the *minimal instances*, those that are legal and do not properly contain any other legal instance [7]. Those global instances do not contain redundant data, and we can concentrate on inconsistency already at their level. In our example, the minimal instances are $Mininst(\mathcal{G}) = \{\{P(a, c), P(d, e), P(a, z), R(z, b)\} \mid z \in \mathcal{U}\}$.

The *minimal answers* to a global query Q are those answers that can be obtained from every minimal instance. In consequence, the certain answers to a query are contained in its minimal answers: $Certain(Q, \mathcal{G}) \subseteq Minimal(Q, \mathcal{G})$. For monotone queries these two sets coincide [7].

Now, let us assume that we have the global FD on P: $A \rightarrow B$. Some of the minimal instances are consistent, e.g. $D_1 = \{P(a, c), P(d, e), R(c, b)\}$, but others are not, e.g. $D_2 = \{P(a, c), P(d, e), P(a, f), R(f, b)\}$. The repairs of system $\mathcal{G}$ are defined as the S-repairs of the minimal instances. For example, D_1 here is its own repair, but D_2 gives rise to two repairs: $\{P(a, c), P(d, e), R(f, b)\}$ and $\{P(d, e), P(a, f), R(f, b)\}$. The consistent answers to a global query Q are those answers that can be obtained from every repair of $\mathcal{G}$ [7].

In order to compute consistent answers from $\mathcal{G}$, we proceed as follows [13, 14]: (1) First the minimal instances of $\mathcal{G}$ are specified as the stable models of a logic program. (2) The repairs of $\mathcal{G}$ are specified as illustrated in Section 2. This constitutes a second layer on top of the program in (1). (3) A third layer is the query program that is run in combination with the other two previous layer. This program contains an $Answer$ predicate that collects the consistent answers, that are computed according to the skeptical stable model semantics of the combined program.

This methodology provably works for first-order queries (and Datalog extensions), and universal ICs combined with (acyclic) referential ICs. As an interesting subproduct we obtain a methodology to compute certain answers to monotone queries (just forget the intermediate repair layer).

In our example, the minimal instances are specified by the program containing the following rules (in addition to the facts $dom(a)$, $dom(b)$, $dom(c)$, $\dots$, $S_1(a, b)$, $S_1(d, e)$, $S_2(a, c)$):

$$\begin{aligned} P(X, Z) &\leftarrow S_1(X, Y), F((X, Y), Z). \\ P(X, Y) &\leftarrow S_2(X, Y). \\ R(Z, Y) &\leftarrow S_1(X, Y), F((X, Y), Z). \\ F((X, Y), Z) &\leftarrow S_1(X, Y), dom(Z), \\ &\qquad choice((X, Y), (Z)). \end{aligned}$$

This program is inspired by the inverse rules algorithm for computing certain answers [26]. Here, F is a functional predicate, whose functionality on the second argument is imposed through the use of the *choice operator* $choice((\bar{X}), (Z))$, that non-deterministically chooses a unique value for Z for each combination of values for $\bar{X}$ [36]. The program with choice can be transformed into a usual normal program with stable model semantics [36]. In our case, there is a one-to-one correspondence between its models and the minimal instances of the integration system [14]. This program not only specifies the minimal instances, but, in combination with a query program, can be also used to compute certain answers to monotone queries. Specification programs can also be produced for case where sources may be *closed* (or *complete*) or *clopen* (or exact) [8].

Now, in order to specify the repairs of $\mathcal{G}$, we can use a program like the one presented in Example 4, as a second layer on top of the previous program.

In a series of papers [17, 16, 42] virtual data integration systems under the *global-as-view* (GAV) approach (global relations are defined as views over the source relations). Different alternative semantics are studied and different classes of global ICs are considered. For example, in their

[4]For the basic notions of virtual data integration we refer to [43].

terminology, the repairs of D_2 would be *loosely sound*, in the sense that, as global instances, they are not legal because the computed views do not extend the given source contents.

In semantic approaches to peer-to-peer data exchange systems [39], peers exchange data according to declarative mappings between them, with the purpose of complementing a peer's data when a local queries posed to peers have to be answered. The exchange of data is governed by theses data mappings, mappings derived by transitivity, and a peer's own local constraints that have to be respected when data from other peers is received [9, 19]. Even more, there may be certain *trust relationships* between peers that should also be taken into account. When inconsistencies are "solved", a peer may decide to trust one peer more than the others that are contributing with data [9].

Acknowledgments: Research supported by NSERC, and CITO/IBM-CAS. L. Bertossi is Faculty Fellow of IBM Center for Advanced Studies (Toronto Lab.). Several contributions and comments by Loreto Bravo are very much appreciated.

References

[1] Arenas, M., Bertossi, L. and Chomicki, J. Consistent Query Answers in Inconsistent Databases. *Proc. ACM Symposium on Principles of Database Systems (PODS)*, ACM Press, 1999, pp. 68–79.

[2] Arenas, M., Bertossi, L. and Chomicki, J. Scalar Aggregation in FD-Inconsistent Databases. *Proc. International Conference on Database Theory (ICDT 01)*, Springer LNCS 1973, 2001, pp. 39-53.

[3] Arenas, M., Bertossi, L., Chomicki, J., He, X., Raghavan, V. and Spinrad, J. Scalar Aggregation in Inconsistent Databases. *Theoretical Computer Science*, 2003, 296(3):405-434.

[4] Arenas, M., Bertossi, L. and Chomicki, J. Answer Sets for Consistent Query Answering in Inconsistent Databases. *Theory and Practice of Logic Programming*, 3(4–5):393–424, 2003.

[5] Barcelo, P. and Bertossi, L. Logic Programs for Querying Inconsistent Databases. *Proc. Practical Aspects of Declarative Languages (PADL 03)*, Springer LNCS 2562, 2003, pp. 208-222.

[6] Barcelo, P., Bertossi, L. and Bravo, L. Characterizing and Computing Semantically Correct Answers from Databases with Annotated Logic and Answer Sets. In *Semantics of Databases*, Springer LNCS 2582, 2003, pp. 1-27.

[7] Bertossi, L., Chomicki, J., Cortes, A. and Gutierrez, C. Consistent Answers from Integrated Data Sources. *Proc. Flexible Query Answering Systems (FQAS 02)*, Springer LNCS 2522, 2002, pp. 71–85.

[8] Bertossi, L. and Bravo, L. Consistent Query Answers in Virtual Data Integration Systems. In *Inconsistency Tolerance*, Springer LNCS 3300, 2004, pp. 42-83.

[9] Bertossi, L. and Bravo, L. Query Answering in Peer-to-Peer Data Exchange Systems. *Proc. EDBT International Workshop on Peer-to-Peer Computing & DataBases (P2P&DB 04)*, Springer LNCS 3268, 2004, pp. 478-485.

[10] Bertossi, L., Bravo, L., Franconi, E. and Lopatenko, A. Fixing Numerical Attributes under Integrity Constraints. *Proc. International Symposium on Database Programming Languages (DBPL 05)*, Springer LNCS 3774, 2005, pp. 262-278.

[11] Bertossi, L., Chomicki, J., Godfrey, P., Kolaitis, Ph., Thomo, A. and Zuzarte, C. Exchange, Integration, and Consistency of Data: Report on the ARISE/NISR Workshop. *SIGMOD Record*, 2005, 34(3):87-90.

[12] Bertossi, L. and Chomicki, J. Query Answering in Inconsistent Databases. In *Logics for Emerging Applications of Databases*. Springer, 2003, pp. 43-83.

[13] Bravo, L. and Bertossi, L. Logic Programs for Consistently Querying Data Integration Systems. *Proc. International Joint Conference in Artificial Intelligence (IJCAI 03)*, Morgan Kauffmann Publishers, 2003, pp. 10-15.

[14] Bravo, L. and Bertossi, L. Disjunctive Deductive Databases for Computing Certain and Consistent Answers to Queries from Mediated Data Integration Systems. *Journal of Applied Logic*, 2005, 3(2):329-367.

[15] Bravo, L. and Bertossi, L. Semantically Correct Query Answers in the Presence of Null Values. *Proc. EDBT International Workshop on Inconsistency and Incompleteness in Databases (IIDB 06)*. To appear in Springer LNCS. Technical Report arXiv:cs.DB/0604076 v1. Posted April 19, 2006.

[16] Cali, A., Calvanese, D., De Giacomo, G. and Lenzerini, M. Data Integration Under Integrity Constraints. In *Proc. Conference on Advanced Information Systems Engineering (CAISE 02)*, Springer LNCS 2348, 2002, pp. 262–279.

[17] Cali, A., Calvanese, D., De Giacomo, G. and Lenzerini, M. On the Role of Integrity Constraints in Data Integration. *IEEE Data Engineering Bulletin*, 2002, 25(3): 39-45.

[18] Calì, A., Lembo, D. and Rosati, R. On the Decidability and Complexity of Query Answering over Inconsistent and Incomplete Databases. *Proc. ACM Symposium on Principles of Database Systems (PODS)*, ACM Press, 2003, pp. 260–271.

[19] Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M. and Rosati, R. Inconsistency Tolerance in P2P Data Integration: An Epistemic Logic Approach. *Proc. International Symposium on Database Programming Languages (DBPL 05)*, Springer LNCS 3774, 2005, pp. 90-105.

[20] Caniupan, M. and Bertossi, L. Optimizing Repair Programs for Consistent Query Answering. *Proc. International Conference of the Chilean Computer Science Society (SCCC 05)*, IEEE Computer Society Press, 2005, pp. 3-12.

[21] Celle, A. and Bertossi, L. Querying Inconsistent Databases: Algorithms and Implementation. In *Computational Logic - CL 2000*, Springer LNCS 1861, 2000, pp. 942-956.

[22] Chomicki, J., Marcinkowski, J. and Staworko, S. Computing Consistent Query Answers using Conflict Hypergraphs. *Proc. Conference on Information and Knowledge Management (CIKM 04)*, ACM Press, 2004, pp. 417 - 426.

[23] Chomicki, J. and Marcinkowski, J. Minimal-Change Integrity Maintenance Using Tuple Deletions. *Information and Computation*, 197(1-2):90-121, 2005.

[24] Dantsin, E., Eiter, T., Gottlob, G. and Voronkov, A. Complexity and Expressive Power of Logic Programming. *ACM Computing Surveys*, 2001, 33(3): 374-425.

[25] Downey, R.G. and Fellows, M.R. *Parameterized Complexity*. Springer, Monographs in Computer Science, 1999.

[26] Duschka, O., Genesereth, M. and Levy, A. Recursive Query Plans for Data Integration. *Journal of Logic Programming*, 2000, 43(1):49-73.

[27] Eiter, T., Fink, M., Greco, G. and Lembo, D. Efficient Evaluation of Logic Programs for Querying Data Integration Systems. *Proc. International Conference on Logic Programming (ICLP 03)*, Springer LNCS 2916, 2003, pp. 163-177. 2916, 2003.

[28] Fagin, R., Kolaitis, Ph. G., Miller, R. J., and Popa, L. Data Exchange: Semantics and Query Answering. *Proc. International Conference on Database Theory (ICDT)*. 207–224, 2003.

[29] Flesca, S., Furfaro, F. Parisi, F. Consistent Query Answers on Numerical Databases under Aggregate Constraints. *Proc. International Symposium on Database Programming Languages (DBPL 05)*, Springer LNCS 3774, 2005, pp. 279-294.

[30] Flum, J. and Grohe, M. Parameterized Complexity Theory. Springer, Texts in Theoretical Computer Science, 2006.

[31] Franconi, E., Laureti Palma, A., Leone, N., Perri, S. and Scarcello, F. Census Data Repair: a Challenging Application of Disjunctive Logic Programming. *Proc. Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 01)*. Springer LNCS 2250, 2001, pp. 561-578.

[32] Fuxman, A. and Miller, R. First-Order Query Rewriting for Inconsistent Databases. *Proc. International Conference on Database Theory (ICDT 05)*, Springer LNCS 3363, 2004, pp. 337-351.

[33] Fuxman, A., Fazli, E. and Miller, R.J. ConQuer: Efficient Management of Inconsistent Databases. *Proc. ACM International Conference on Management of Data (SIGMOD 05)*, ACM Press, 2005, pp. 155-166.

[34] Gelfond, M. and Lifschitz, V. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 1991, 9:365–385.

[35] Greco, G., Greco, S. and Zumpano, E. A Logical Framework for Querying and Repairing Inconsistent Databases. *IEEE Transactions on Knowledge and Data Engineering*, 15(6):1389–1408, 2003.

[36] Giannotti, F., Greco, S., Sacca, D. and Zaniolo, C. Programming with Non-determinism in Deductive Databases. *Annals of Mathematics and Artificial Intelligence*, 1997, 19(1-2):97-125.

[37] Grieco, L., Lembo, D., Rosati, R. and Ruzzi, M. Consistent Query Answering under Key and Exclusion Dependencies: Algorithms and Experiments. *Proc. ACM International conference on Information and Knowledge Management (CIKM 05)*, ACM Press, 2005, pp. 792-799.

[38] Grohe, M. Parameterized Complexity for the Data-base Theorist. *SIGMOD Record*, 2002, 31(4):86-96.

[39] Halevy, A., Ives, Z., Suciu, D. and Tatarinov, I. Schema Mediation in Peer Data Management Systems. *Proc. Intenational Conference on Data Engineering (ICDE 03)*, IEEE Computer Society Press, 2003, pp. 505-518.

[40] Immerman, N. *Descriptive Complexity*. Graduate Texts in Computer Science. Springer, 1999.

[41] Kolaitis, Ph. Schema Mappings, Data Exchange, and Metadata Management. *Proc. ACM Symposium on Principles of Database Systems (PODS 05)*, ACM Press, 2005, pp. 61-75.

[42] Lembo, D., Lenzerini, M. and Rosati, R. Source Inconsistency and Incompleteness in Data Integration. In *Proc. International Workshop Knowledge Representation meets Databases (KRDB 02)*, CEUR Electronic Workshop Proceedings, 2002.

[43] Lenzerini, M. Data Integration: A Theoretical Perspective. *Proc. ACM Symposium on Principles of Database Systems (PODS 02)*, ACM Press, 2002, pp. 233-246

[44] Lopatenko, A. and Bertossi, L. Complexity of Consistent Query Answering in Databases under Cardinality-Based and Incremental Repair Semantics. Technical Report arXiv:cs.DB/0604002 v1. Posted April 2, 2006.

[45] Miltersen, P.B., Subramanian, S., Vitter, J.S. and Tamassia, R. Complexity Models for Incremental Computation. *Theoretical Computer Science*, 1994, 130(1):203-236.

[46] Niedermeier, R. and Rossmanith, P. An Efficient Fixed-Parameter Algorithm for 3-Hitting Set. *Journal of Discrete Algorithms*, 2003, 1(1):89-102.

[47] Weber, V. and Schwentick, T. Dynamic Complexity Theory Revisited. *Proc. Annual Symposium on Theoretical Aspects of Computer Science (STACS 05)*, Springer LNCS 3404, 2005, pp. 256-268.

[48] Wijsen, J. Condensed Representation of Database Repairs for Consistent Query Answering. *Proc. International Conference on Database Theory (ICDT)*, pages 378–393. Springer-Verlag, LNCS 2572, 2003.

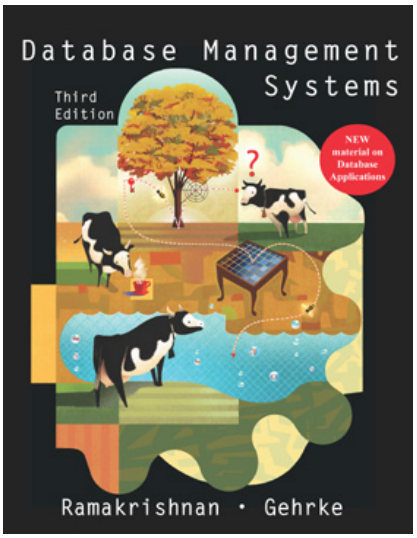
Raghu Ramakrishnan Speaks Out

on Deductive Databases, What Lies Beyond Scalability, How He Burned Through $20M Briskly, Why We Should Reach Out to Policymakers, and More

by Marianne Winslett

Raghu Ramakrishnan
http://www.cs.wisc.edu/~raghu/

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 Department of Computer Science at the University of Illinois at Urbana-Champaign. I have here with me Raghu Ramakrishnan, who is a Professor of Computer Sciences at the University of Wisconsin-Madison. Raghu's research currently focuses on data retrieval and integration, analysis, and mining. Raghu was a founder of QUIQ, a company that developed a collaborative customer support facility. Raghu has received the National Science Foundation's Presidential Young Investigator award, a Packard Fellowship, and a SIGMOD Contributions Award for founding and maintaining DBWorld. Raghu is an ACM Fellow; he is a coauthor of the popular Database Management Systems *textbook; and he is also the current chair of ACM SIGMOD. Raghu's PhD is from the University of Texas at Austin. So, Raghu, welcome!*



Raghu, let's get the most important question out of the way first. What does the cover of your textbook, Database Management Systems*, mean?*

It's a depiction of how queries are processed. The cows are users, from Wisconsin of course, with the same number of cows as the number of editions. [Wisconsin is well known for its cheese.] The question marks are queries. When you have a query, the arrows show you what happens. You go to the root of a B-tree, and that directs you to a leaf, where you find a key. You take that key over to a relational table, to find the records you need. Then you put the records into a buffer pool, where you cook them, using relational operations. That, in a nutshell, is what database systems are all about: carrying out computation across memory hierarchies and moving data back and forth in a clever way.

You cut your teeth on research into deductive databases, with the Coral project. Deductive databases have been out of fashion for many years, but you had a paper in SIGCOMM 2005 entitled "Declarative Routing: Extensible Routing with Declarative Queries." Does this signal a renaissance for deductive database research?

The paper originated because Joe Hellerstein asked me to be involved in this work. Joe and the Telegraph group at Berkeley and at Intel have been looking extensively at how to use recursive rules inside of network routers, to do all kinds of cool things. Joe is fond of saying that maybe Oracle and the other database companies will support deductive databases, or perhaps Cisco will have a recursive query processing product soon.

As another example, Serge Abiteboul and his colleagues had a paper recently where recursion was driving an application [Serge Abiteboul, Zoë Abrams, Stefan Haar, and Tova Milo, "Diagnosis of asynchronous discrete event systems: Datalog to the rescue!", PODS 2005, pp. 358-367]. Their proposed system had some of the elements of the trust policy languages that you were talking about earlier, with distributed rules executing---essentially, Datalog with a fixed point. In that paper, essentially they were simulating a certain class of Petri nets of finite state machines in a highly distributed fashion.

Of course, the trust policy work that you and I talked about yesterday also uses Datalog. And when I was at Microsoft some time ago, someone there wanted a copy of Coral because they were using recursive rules in their debugging work. Their work involved graphical representations of programs, and they wanted to borrow an open source recursive language.

So it turns out that there are indeed a lot of recursive applications. Enough to sustain an industry? I don't know about that. It does seem that there is a growing resurgence of interest. Maybe that is because of the natural presence of recursion across nodes in the internet; maybe it is because of languages like XML (XPath has recursion); or maybe it is because of natural delegation in things like policies for security and access control. There is a lot of recursion latent in the world and I think we are finally coming to applications that are tapping into it.

Since we are talking about old things that are new again, let me ask about data cubes. Are they back in style again?

I would like to see them come back in style. If you look at data mining, traditionally what database people have brought to the table is scalability. Given an algorithm that works well with a small data set, database people have figured out how to make it work with much larger data sets. Database people have done this for clustering, classification, frequent item sets. But I think we have a whole lot more to bring to the table than just scalability. We understand notions like compositionality, and we understand things like how to structure a space for exploratory analysis. That is what the multidimensional data model and data cubes are all about. How do you take this set of concepts and combine them with predictive models or other kinds of tools borrowed from statistics and machine learning? In the case of machine learning, a standard research focus early on was how to take a given "case" and analyze it: should I give this person a loan or not? In the case of exploratory analysis, the emphasis has been not so much on an individual case, but rather on understanding the data set as a whole: in what parts of the world and in what periods of time did loan decisions have a certain bias? To answer these kinds of questions, you can employ the power of the exploratory analysis tools that we database people have developed, and combine them very profitably with the kinds of summarization and predictive tools that the machine learning and statistics communities have been developing. I think this could lead to a hybrid set

of tools, a hybrid set of approaches to exploratory analysis, that are conceptually more powerful. In other words, we would not just be scaling up existing tools developed in these related research communities, but instead developing a conceptually broader and entirely different class of hybrid tools for analyzing the enormous amounts of data coming at us.

It seems to me that you have now gone full circle in your research interests---back to where you started. Are you all done now, or are you going to go around the circle again?

Hopefully not the same circle.

A spiral?

Hopefully not a downward one. I enjoy going around and around in circles, I think, so I will probably do that for a while.

The 1989 Laguna Beach report on future directions in database research was quite negative regarding deductive database research. In hindsight, what do you think about such reports and their impact on deductive database research?

I must confess that back when I first came across the report, I was not a happy camper. Over the years, my reaction has mellowed. I think I do understand the value of having senior researchers in a field offer their opinions to help junior researchers in making their choices as to what to work on. But that said, I still have mixed feelings about the value of reports like this. On the whole, I would like to see emphasis on directions that people think have promise, rather than emphasis on things that one or another group of people feel ought not to be emphasized.

Why is that? I could see both of those playing a role: do this, don't do that.

When you think something is important enough to spend your time on, and have invested a significant fraction of your time and energy, I like to think that you have given enough thought to that choice. Now, if you think that something is *not* important, you have actively cast your vote by simply working on something else that you believe *is* important. By actively downplaying a topic that others might consider important, I think you could be doing a disservice to something that you have not thought about as deeply as those who are committed to that particular vision. So I would rather see you thoroughly focus your energies on promoting that which you have given the most thought to and invested the most in, because that is probably where you have the most insight and the most passion, and where you can have the most legitimate kind of influence.

How have database vendors made use of the many years of deductive database research in improving the performance of SQL3 queries that involve recursion?

I think that there are two areas, at least. First, the idea of magic sets rewriting has helped in dealing with correlated queries. When you have nested queries with correlation, magic sets rewriting turns out to be a very useful tool for dealing with correlation in a set-oriented fashion. I believe several venders have used this to get better performance on the TPC-D benchmark. Second, incremental view maintenance became a very hot area in the 90s. Virtually every vendor supports some version of this, and most of these techniques generalize and build upon some form of semi-naive evaluation. Both of these core techniques, magic sets rewriting and semi-naive evaluation, were developed initially in the deductive database community.

What do you think database theoreticians should be working on today?

In accordance with my earlier comments on how to answer questions like this, let me speak strictly from the perspective of the areas I have worked in. I have looked lately at information extraction from text. I think there is a lot of work to be done on handling data with associated uncertainty and handling probabilistic reasoning, not just in a query setting, but also synthesizing ideas from the statistical learning and the statistical analysis literatures. Fundamental practical problems like deduplication can have elegant mathematical frameworks, too. In fact, I think all of these topics do have active ongoing work of a theoretical nature. I think theoreticians have recognized that these are areas of opportunity, and they are working on them. Another broad area of opportunity is social network analysis, which seems to be popular in the web community. And simply because of the scale of the data, I figure that counts as databases, although perhaps it is not mainstream databases (yet!).

Raghu, during the course of your career, you have moved from pure theory to systems research, and from academia to a startup and back again. What led you to make these changes?

I have always had an interest in studying things that involve some kind of real application. Even in the earliest days, when my work was about recursive queries, it was to be included in an implemented system. Initially, when I was a graduate student, the target was the LDL system, and then later as I started my own research, it was the Coral system. So most of the theory I did was related to some ongoing practical effort that I was involved in.

Over the years, I have come to build and reason about systems more often, whenever an opportunity offers itself. But I don't see this as a change, really. While the percentages may have shifted over time, the basic interplay between systems and theory has been part of what I have done all along.

What about your startup company, QUIQ?

The startup was something I just could not resist. There was an opportunity to really do something outside of academia, to actually build a product that billions and billions of people would use. There was also an opportunity to work together with my brother, who had graduated with an MBA from Stanford: the opportunity to have a business person and a technical person work together.

Those were boom times, money seemed to grow on trees, and we actually raised about $20 million over the course of a couple of years. It was great fun building the company, and we did some pretty cool things similar to Yahoo Answers today. We were doing that for Ask Jeeves back in 1999, with a million different alerts being monitored continuously for Ask Jeeves alone. We had clients like Compaq, Sun, National Instruments. The idea of using mass collaboration in the context of technical support and question answering was a really very intriguing concept to try and take all the way to the point of commercialization. The end game was less enjoyable.

Tell us about the end game.

The bottom line is that we worked our way through $20 million a little more briskly than I would have liked. It took us more time than it should have to translate our technology into a coherent business strategy. The sad part is that we eventually did come up with that strategy. Compaq was using our product extensively at the end, they were paying us very significant money, and they were very happy with what they were getting in return. Several large companies were already our clients, but we were caught in a bind where we were running out of money at around

the same time that we were trying to close a bunch of deals. We were not able to close them fast enough, given the economic downswing then. And a few other things happened. We had a key deal with Apple which fell apart when the group at Apple got fired, after we had started working for them. And the venture capitalists---at that point, shall we say, they did not want to invest in that climate. And all said and done, we had to wind things down and sell the company at a level that made no one very happy. There were earlier opportunities when we could have exited much more gracefully and profitably.

While building the company was enormously exciting, on the downward path, we had to tighten our belts and let people go. Those were some of the hardest things I have done.

All in all, it was a unique experience and something that I think I wouldn't trade.

How was the transition back to academia?

Hard. I think that the venture capitalists would have like the first couple of grant proposals I wrote, but my peers didn't. My whole value system was skewed. I was looking for things that made business sense, as opposed to academic novelty. And it took me a while to readjust my senses.

Did you have the reverse problem when you first got to QUIQ---would NSF have liked your proposals?

I did tell you we blew through $20 million briskly, without making much money, so I will let you draw your own conclusions.

In the beginning, data mining research seemed to have a home in the database world, but now it seems to have evolved into a separate field with three major conferences of its own. What should the relationship be between the two fields?

I think it is healthy for the two fields to have a complementary relationship like they do now. Data mining involves more than databases; machine learning and statistics are integral components of data mining. It is good for the data mining community to have their own identity; to try and form a set of foundational ideas that are unique to data mining, that are at the intersection of and distinct from each of the contributing fields.

At the same time I very much hope that we continue to see data mining papers in the main database conferences, that we see database researchers at least attending the occasional data mining conference, and that we don't see a split between the communities. Obviously, this is going to be beneficial for data mining, but equally, I think, the database field in the future is going to be largely concerned with how to make sense of data. A lot of what is happening in data mining is very relevant to this goal. For example, if you are doing automatic tuning at virtually any level of the system, you can instrument your system and apply learning techniques to help in the tuning. What other techniques are appropriate in your setting? Who might you collaborate with? I would like to us take advantage of these synergies.

Raghu, recently you have become interested in data privacy and security. This kind of research brings in new kinds of considerations that database researchers are not used to thinking about: legal, ethical, and public policy. For example, if I publish a join algorithm that I say is 20% faster, no one really cares if I am wrong. But if I say I have a new data anonymization technique that preserves privacy 20% better, that is a scarier promise: the public is going to think that it

means something. Have you come to grips with this new dimension, and how are you going to handle it?

It's a very good question. I spent some time recently working on a paper on data mining and the law, together with Deirdre Mulligan, who is a professor of law at Berkeley, and Chris Clifton from Purdue. It was a real learning experience. We technical researchers are developing tools that can be used very invasively. I don't think the appropriate response to that is to say that we shouldn't develop such tools. Those tools are going to be developed, by someone else if not by us. At the same time, I don't think the appropriate response is to say, "Here's a technique, and all I did was develop it, and I am no longer responsible for what someone else may do with it."

Did you hear in the news today that the government has subpoenaed all Google searches for the past k *years?*

Yes. Google seems to be at the cutting edge of so many different legal questions.

But they are also about scale, just like we are.

They are about scale, they are about copyright. Consider the topic of databases on the web: ownership, legal rights, privacy---these are all related, you cannot entirely separate them. Ownership is at the heart of many of these questions. Whether you are talking about ownership of copyright, or thinking of ownership of my private information, this whole thing is a gray area on the web.

But to get back to your original question, I think we database researchers should be thinking a great deal about issues such as how to characterize the extent to which different tools can be used to achieve certain kinds of compromises of privacy and security in different settings. This doesn't mean we set policy, but we have to provide guidance to those who do set policy. And we have to provide tools to help enforce more flexible policies.

We need to be thinking about this in terms of the algorithms we develop, or the theorems we prove: don't just prove theorems about how fast something is, prove theorems about how far you can go in, e.g., cracking a certain kind of anonymization. Not only do we have this new agenda research-wise, I also think we have a burden to do something that we have traditionally not done. We need to be involved socially and legislatively, to influence people who set policy. We need to inform them, because they may not be fully aware of the kinds of risks associated with the data gathering that is going on today, and the tools that are available to subsequently analyze that data. We may not be aware of all the data gathering that is going on, but certainly we are aware of some of the possibilities. We are aware of many of the tools that are available or can be easily developed. I think we need to become more socially active.

Traditionally, we haven't been very good at pursuing that engagement. So how is it going to happen? Who is going to do it?

It is going to be us, although we have not taken on this particular gauntlet in the past---

I mean which *us?*

You and me: the people who have tenure in the universities, leadership positions in industry, us; not the young ones who have other concerns that are more immediately pressing for them. The

older people like us have both the time and the perspective to step back and look at not just the immediate research we do, but the consequences of that research. We ought to be doing that.

How did you end up in database research?

This is an interesting question. I went to the University of Texas and took a course on databases with Hank Korth. I really enjoyed it. I really don't know how I came to knock upon Avi Silberschatz's door, but I did, and we started working together. My first paper was on concurrent logic programs; it appeared in POPL, and I decided that it would not be my thesis area. Avi then pointed me to some work on data modeling. And then along came MCC and I found myself in a position where there was this entire new field of deductive databases emerging. I had the opportunity to work with some brilliant guys, excellent people, like Catriel Beeri, Francois Bancilhon, Carlo Zaniolo, and others. And I had exactly the right background; I knew about logic programs, I knew about databases. So I just jumped in there, and after that, I just kept going.

How was your schedule as a graduate student in Austin?

Well, that was back in the days when you had shared mainframes. Working during the day was painfully slow. Parking wasn't very good, either, so I took to working after dinner and going to sleep after breakfast. With that approach, I got excellent parking, and my machine was as fast as my machine is today. Some people did look at me squinty eyed because I ate cereal in the evenings. When eventually Bancilhon went back to France, his parting comment was, "Finally Raghu and I will be awake at the same time."

What is the spiciest food you have ever eaten?

I once made the mistake of going into a Thai restaurant and asking them to make it very spicy. I like spicy food, but never, ever will I go into a Thai restaurant and ask for very spicy again.

Can you describe it to us? How spicy was it?

Too painful to remember.

The past couple of decades have been hard for many professional associations, which have seen large drops in membership. Computer science associations have not been immune to this trend. What do you see as the role of ACM SIGMOD in the life of database researchers over the next ten years?

First of all, there is also an opportunity here, especially for information technology, for databases. Countries like India and China are exploding in terms of the number of professionals. Our enrollment of members from those countries is incredibly low. We need to be much more aggressive in getting members globally and retaining them beyond the first year.

And what will you offer to the people in India and China that will make them want to join?

Networking and awareness of what we as a community are doing. These are the same benefits that we get here: staying in the loop.

Can't they get that off the web?

I see the web as an opportunity. It used to be that you got the most benefit out of these kinds of memberships if you were in a position to physically attend the conferences, which were mostly in the US and occasionally in Europe. The web makes it possible to capture the conferences in video and make them available on the web, and this takes money. Part of that money comes from memberships. We have to figure out a business model, because at the end of the day ACM SIGMOD is a business. Our focus is not on profits, but we have to stay above water; we are not subsidized by anyone. We need to figure out how we can take all the things that are going on, such as conferences, symposiums, and other events, and make them accessible to people in our field, no matter where they are geographically. That's the benefit that SIGMOD members get, and hopefully they consider it worthwhile enough to pay for, and, in turn, to let us make those benefits generally available.

We also need to have our conferences in distant parts of the world. I think we are doing that. SIGMOD has made a very conscious effort to reach out and relocate elsewhere. I hope to see that continue. I think organizations like SIGMOD can also take a leadership role in bringing together communities outside of SIGMOD that share common interests, such as SIGIR, SIGKDD, and other SIGs where there would be benefit in researchers coming together. Maybe we can be creative and work together with those SIGs, to create opportunities for cross fertilization.

From among your past papers, do you have a favorite piece of work?

I have two. First, the magic sets paper I wrote is one of them, simply because I spent six months trying to figure out how to generalize the version that Bancilhon, Sagiv, Maier, and Ullman had written, and the solution came in the space of five minutes over coffee. There was an "Aha!" moment that I will always remember. As my other favorite, I did some work on dealing with sorted relations (sequences) that appeared in SSDBM and is relatively little known. It ended up influencing window functions in SQL99 quite significantly. I have always had an interest in streams and sequences, and this was the piece of work that had the most influence, so I have a fond spot for it.

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

I would probably have lunch with my colleagues more often.

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

I would take some smart pills.

People say that you have a great sense of humor. Do you have a joke to share with us?

I should tell you the story about Bill Gates and the Indian entrepreneur. Long ago, an Indian entrepreneur came to visit Redmond, and told Bill, "You really ought to go to India and hire people, because that is where the action is." But Bill wasn't interested. He said, "No, come with me." He gave the Indian guy a spade and asked him to dig. The entrepreneur dug, and dug, and dug. Up came a cable, and Bill said, "*Cable!* That's where the action is." (This was back during the time when Bill was trying to buy cable companies.) The Indian entrepreneur was very upset and went back home. Eventually, though, Bill saw the light, and went to India to recruit. (Microsoft was one of the very first US companies to recruit directly in India.) His Indian entrepreneur buddy gave Bill a spade, and said, "Dig." Bill had to dig, to return the favor. It's hot in India, and five minutes later, Gates was sweating. After ten minutes, fifteen minutes, he

was knee deep, and he's not exactly a spring chicken, so he was getting very tired. He looked up, and the Indian told him, "*Wireless!* That's where the action is."

Thank you, Raghu, for talking with us.

You are very welcome.

Changes to the *TODS* Editorial Board

Richard Snodgrass

rts@cs.arizona.edu

Mike Franklin left the Editorial Board on March 1 after five years of very active service, handling an even two dozen *TODS* submissions. I especially appreciate Mike's cogent advice on the many policy issues that arose over the last five years.

I'm very pleased to announce the appointment of a new Associate Editor.

Jan Van den Bussche is currently active in the design and analysis of data models and query languages; in data mining; and in scientific databases.

Actually, Jan was appointed with the last batch of four (introduced in the December 2005 issue of *SIGMOD Record*), but he wanted to start after PODS, for some reason...

Thanks to Jan and indeed all on the *TODS* Editorial Board[1] for their willingness to invest their valuable time to handle manuscripts.

Finally, let me remind you that the papers in upcoming issue(s) are available on the web[2] as are the extensive referee rights,[3] probably the most extensive for a database journal, and various publication statistics,[4]. This wonderful editorial board has managed to meet its five-month maximum turnaround promise for almost two years now (not that long ago the *average* was over five months).

[1] http://www.acm.org/tods/Editors.html
[2] http://www.acm.org/tods/Upcoming.html
[3] http://www.acm.org/tods/Referees.html
[4] http://www.acm.org/tods/TurnaroundTime.html

CALL FOR PARTICIPATION

MobiDE 2006: Fifth International ACM Workshop on Data Engineering for Wireless and Mobile Access

June 25, 2006, Chicago, IL
In conjunction with SIGMOD/PODS 2006

http://db.cs.pitt.edu/mobide06

Important Dates:

May 15: Early registration

June 25: Workshop

We invite you to participate in the ACM MobiDE 2006 workshop, the fifth in a successful series of workshops that bring together the data management, wireless networking, and mobile computing communities.

The workshop has received a substantial number of submissions spanning a wide range of exciting topics in mobile and wireless data management. As in the past, the workshop will be organized in a manner that fosters interaction and exchange of ideas among participants. Thus, in addition to presentations of accepted papers, the program will be designed to accommodate discussions among participants. The detailed technical program and invited talks will be posted on the workshop website after May 15th.

MobiDE 2006 is sponsored by ACM SIGMOD and held in cooperation with ACM SIGMOBILE. The industrial sponsors are Sereniti, HP Labs, and Microsoft Research.

You can register for the workshop and obtain hotel information through the SIGMOD 2006 website: `http://www.regmaster2.com/sigmod2006.html`

Program Committee:

A. E. Abbadi, UC Santa Barbara, USA
W. Aref, Purdue U., USA
M. Balazinska, U. Washington, USA
S. Banerjee, HP Labs, USA
C. Becker, U. Stuttgart, Germany
U. Cetintemel, Brown U., USA
N. Davies, Lancaster U., UK
M. Dunham, SMU, USA
T. Hara, Osaka U., Japan
R. Jain, Google, USA
V. Kalogeraki, UC Riverside, USA
D. Katsaros, U. Thessaloniki, Greece
D. L. Lee, U. of Sci &Tech, Hong Kong
M. L. Lee, National U. of Singapore
W. Lehner, TU Dresden, Germany
L. Liu, Georgia Tech, USA
S. Madria, U. Missouri, Kansas City, USA
P. J. Marron, U. Stuttgart, Germany
D. Nicklas, U. Stuttgart, Germany
M. Papadopouli, UNC at Chapel Hill, USA
E. Pitoura, U. Ioannina, Greece
C. Lucia, U. Grenoble, France
G. Samaras, U. Cyprus, Cyprus
J. Sander, U. Alberta, Canada
B. Seeger, U. Marburg, Germany
J. Su, UC Santa Barbara, USA
Y. Tao, City U. of Hong Kong, Hong Kong
V. Tsotras, UC Riverside, USA
A. K H Tung, National U. of Singapore
W. G. Yee, IIT, Chicago, USA
V. Zadoroshny, U. Pittsburgh, USA
A. Zaslavsky, Monash U., Australia
D. Zhang, Northeastern U., USA

Workshop Chairs:

Vijay Kumar
U. of Missouri-Kansas City

Alexandros Labrinidis
U. of Pittsburgh

Program Chairs:

Panos K. Chrysanthis
U. of Pittsburgh

Christian S. Jensen
Aalborg U.

Publicity Chair:

Magdalena Balazinska
U. of Washington

Steering Committee:

Sujata Banerjee, HP Labs
Ugur Cetintemel, Brown U.
Mitch Cherniack, Brandeis U.
Panos K. Chrysanthis, U. of Pittsburgh
Alexandros Labrinidis, U. of Pittsburgh
Evaggelia Pitoura, U. of Ioannina

Sereniti
NEXT GENERATION BROADBAND SERVICES FOR THE CONNECTED HOME

Microsoft Research