## SIGMOD Officers, Committees, and Awardees (continued)

### SIGMOD Edgar F. Codd Innovations Award

*For innovative and highly significant contributions of enduring value to the development, understanding, or use of database systems and databases*. Until 2003, this award was known as the "SIGMOD Innovations Award." In 2004, SIGMOD, with the unanimous approval of ACM Council, decided to rename the award to honor Dr. E.F. (Ted) Codd (1923 - 2003) who invented the relational data model and was responsible for the significant development of the database field as a scientific discipline. Recipients of the award are the following:

| | | |
|---|---|---|
| Michael Stonebraker (1992) | Jim Gray (1993) | Philip Bernstein (1994) |
| David DeWitt (1995) | C. Mohan (1996) | David Maier (1997) |
| Serge Abiteboul (1998) | Hector Garcia-Molina (1999) | Rakesh Agrawal (2000) |
| Rudolf Bayer (2001) | Patricia Selinger (2002) | Don Chamberlin (2003) |
| Ronald Fagin (2004) | Michael Carey (2005) | Jeffrey D. Ullman (2006) |
| Jennifer Widom (2007) | Moshe Y. Vardi (2008) | Masaru Kitsuregawa (2009) |

### SIGMOD Contributions Award

*For significant contributions to the field of database systems through research funding, education, and professional services*. Recipients of the award are the following:

| | | |
|---|---|---|
| Maria Zemankova (1992) | Gio Wiederhold (1995) | Yahiko Kambayashi (1995) |
| Jeffrey Ullman (1996) | Avi Silberschatz (1997) | Won Kim (1998) |
| Raghu Ramakrishnan (1999) | Michael Carey (2000) | Laura Haas (2000) |
| Daniel Rosenkrantz (2001) | Richard Snodgrass (2002) | Michael Ley (2003) |
| Surajit Chaudhuri (2004) | Hongjun Lu (2005) | Tamer Özsu (2006) |
| Hans-Jörg Schek (2007) | Klaus R. Dittrich (2008) | Beng Chin Ooi (2009) |

### SIGMOD Jim Gray Doctoral Dissertation Award

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

- **2006** *Winner*: Gerome Miklau, University of Washington
  *Runners-up*: Marcelo Arenas, Univ. of Toronto; Yanlei Diao, Univ. of California at Berkeley.
- **2007** *Winner*: Boon Thau Loo, University of California at Berkeley
  *Honorable Mentions*: Xifeng Yan, UIUC; Martin Theobald, Saarland University
- **2008** *Winner*: Ariel Fuxman, University of Toronto
  *Honorable Mentions*: Cong Yu, University of Michigan; Nilesh Dalvi, University of Washington.
- **2009** *Winner*: Daniel Abadi (advisor: Samuel Madden), MIT
  *Honorable Mentions*: Bee-Chung Chen (advisor: Raghu Ramakrishnan), University of Wisconsin at Madison; Ashwin Machanavajjhala (advisor: Johannes Gehrke), Cornell University.

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

[Last updated on December 31, 2009]

# Editor’s Notes

Welcome to the December 2009 issue of SIGMOD Record. This is my last issue as SIGMOD Record Editor; more on this later.

We begin the issue with a welcome article from the new Vice-Chair of SIGMOD, Christian Jensen.

The one regular article of this issue, by Van Cappellen, Liu, Melton, and Orgiyan, is about XQJ, the XQuery Java API, which provides a standard API for XQuery engines to declaratively access and manipulate XML data from different sources.

We continue with two articles in the **Surveys Column** (edited by Cesar Galindo-Legaria). The first one, by Whang, Song, Kim, and Lee, reviews the requirements as well as existing systems and research prototypes of Ubiquitous Database Management Systems, i.e., DBMSes that can be used in small mobile devices, such as cellular phones. The second article, by Sakr and Al-Naymat, surveys the different relational alternatives to storing and querying RDF data.

We continue with two articles in the **Systems and Prototypes Column** (edited by Magdalena Balazinska). The first article, authored by Bercovitz, Kaliszan, Koutrika, Liou, Parameswaran, Venetis, Zadeh, and Garcia-Molina, is entitled “*Social Sites Research Through CourseRank*”. This paper describes the CourseRank system that was demonstrated during SIGMOD 2009 and received the best demo award. The second article, by Lang, Patel, and Naughton is entitled “*On Energy Management, Load Balancing and Replication*” and investigates opportunities and challenges in energy-aware computing, which is becoming an increasingly important dimension of any real system deployment.

We continue with three articles in the **Reports Column** (edited by Brian Cooper). First is the *Report of the Third International Workshop on Personalized Access, Profile Management, and Context Awareness in Databases (PersDB 2009)*, written by Amer-Yahia and Koutrika. Second is the *Report on the Second Workshop on Very Large Digital Libraries (VLDL 2009)*, written by Manghi, Pagano, and Ioannidis. Third is the *Report on the Fifth International Workshop on Networking Meets Databases (NetDB 2009)*, written by Loo and Saroiu.

We close the issue with multiple **Calls for Participation**, in connection with the 2010 SIGMOD/PODS Conference to be held in Indianapolis, IN, USA, June 6-11, 2010:

- **SOCC 2010**: First ACM Symposium on Cloud Computing (June 10 & 11, 2010)
- **DBMe 2010**: DataBase Mentoring Workshop at SIGMOD 2010 (June 11, 2010)
- **DBTEST 2010**: Third International Workshop on Testing Database Systems (June 7, 2010)
- **IDAR 2010**: Ph.D. Workshop on Innovative Database Research (June 11, 2010)
- **WANDS 2010**: First International Workshop on WorkflowApproaches to New Data-centric Science (June 6, 2010)
- **WebDB 2010**: 13th International Workshop on the Web and Databases (June 6, 2010)

As I mentioned above, this is my last issue as SIGMOD Record Editor. It has truly been an honor and a great ride, but I had to step down, because of my new duties as SIGMOD Secretary/Treasurer. I am extremely happy to pass the torch to **Ioana Manolescu**, at INRIA. I am confident that Ioana will do a marvelous job as SIGMOD Record Editor, and my first interactions with her have only reinforced my initial opinion.

Writing my last Editor's Notes gave me a chance to pause and reflect on the last three years, firstly on whom to thank, secondly on what we did well and what we did badly (and need to change), and finally, on what would be the advice to pass on to Ioana (and the rest of the community).

First of all, I would like to thank the Associate Editors of SIGMOD Record. The issues would only be a small fraction of what has actually been published had it not been for them. I strongly believe that the multitude of columns and topics is the "secret sauce" that makes SIGMOD Record an interesting read. Brian, Cesar, Denilson, Magda, Marianne, Leonid, Ugur and also Andrew, Jim, and Len, thank you all very much for your hard work. Many thanks also go to the SIGMOD Executive Committee for their help and advice throughout the years, and also to the ACM Headquarters Staff in general and to Julie Goetz in particular, for all her help, patience, and perseverance. Last, but not least, a special thank you to all the people who helped in the review process and are listed below (including my students in the Advanced Data Management Technologies Lab at the University of Pittsburgh).

Secondly, the good things. We established a customized paper submission web site to manage the entire review workflow (`http://db.cs.pitt.edu/recess`). There were two special issues: the June 2008 issue for the Jim Gray Tribute (many thanks to Donna, Pat, Paula, Alan and Vicky) and the December 2008 issue with the special section on Managing Information Extraction (many thanks to AnHai, Luis, Raghu and Shivakumar). We revived the Systems and Prototypes column, and introduced two new columns: the *Open Forum* column, to present (meta-)ideas about non-technical issues and challenges of interest to the entire community, and the column to celebrate the *40-year anniversary of the relational model*. Finally, we introduced a twitter feed (`http://twitter.com/sigmodrecord`), to announce when new issues are published online.

Thirdly, the bad things. Unfortunately, time has been the ultimate frontier on many occasions. The most prominent one is the backlog that has been created in the reviewing of the research papers. I do want to apologize profusely to all the authors whose papers were delayed. The good news is that by the time this issue is printed, I expect all authors to have been notified of the status of their papers. I also expect for the pipeline to be emptied within the next two issues (March and June 2010); I will stay on as an associate editor, to wrap up these papers. The even better news is that, going forward, we are taking major steps to avoid the possibility of a backlog. In particular, we are changing the structure of the SIGMOD Record editorial board to include 2-3 additional associate editors that will help with the reviewing process of the regular research articles. In addition, Ioana will soon be announcing an editorial policy for SIGMOD Record, to make it clear what types of papers are appropriate for publication (i.e., to allow for some self-selection in the process).

In closing, I would like to urge the broader data management community to continue to support SIGMOD Record by:

- reading it (cover to cover), online or in hard-copy
- contributing articles (in any of the columns)
- citing SIGMOD Record papers, as appropriate
- lastly, but not least: reviewing papers, as requested by Ioana or the associate editors. Almost all papers are short (6 pages), so there is no big reason why we cannot decrease the turn-around times.

Thank you.

Alexandros Labrinidis, Former Editor
May 2010

## External Reviewers for SIGMOD Record 2007 - 2009

Lory Al Moakar
Nicolas Anciaux
Walid Aref
Omar Boussaid
Michael Cafarella
Monique Chang
Panos K. Chrysanthis
Sun Chung
AnHai Doan
Johann-Christoph Freytag
Shenoda Guirguis
Joachim Hammer
Panagiotis Ipeirotis
James Joshi
Nodira Khoussainova
Birgitta Koenig-Ries
Hank Korth
Yannis Kotidis
YongChul Kwon
Adam Lee
Jayant Madhavan
David Maier
Panayiotis Neophytou
Thomas Neumann
Jignesh Patel
Evaggelia Pitoura
Huiming Qu
Michael Rys
Pavel Shvaiko
Yannis Sismanis
Vassilis Tsotras
Jie Xu
Vladimir Zadorozhny

# Welcome Message to New (and Continuing) SIGMOD Members

On behalf of ACM SIGMOD's Executive Committee, it is a pleasure to be writing to you for the first time since I became Vice-Chair and thus assumed responsibility for membership issues.

SIGMOD is one of the largest ACM Special Interest Groups. As a member of SIGMOD, you are part of an organization with more than 2,800 members from all across the globe, whose goal is to promote research and technological advancement in the field of data, information, and knowledge management.

We live in an age of data explosion, and SIGMOD holds the potential to play a pivotal role in facilitating and promoting the development of technologies that enable us to harness and benefit from all the data available. To realize this potential, we need to join forces, to work together, and to makes sure that everybody's contributions count. In particular, we need the out-of-the-box thinking of new members of our community such as students and members with non-database backgrounds.

The SIGMOD website (www.sigmod.org) contains the latest information related to the community. It includes descriptions of the benefits associated with the different kinds of membership, it contains contact information, and much other content.

The member benefits include various kinds of content. Our philosophy is to make available content accessible online to the extent possible, thus avoiding expensive and environmentally unfriendly media and paper printing and shipping. However, due to copyright difficulties, not all available content can be provided online. As a result of this, we have different types of membership that provide content on different media.

The ACM SIGMOD Digital Symposium Collection (DiSC) returns in 2010, starting with a two-volume set, DiSC'07/08. DiSC took a break for two unrelated reasons. First, the software used to produce DiSC met an untimely end, resulting in new software having to be written. Second, as times change and business models evolve, issues regarding intellectual property have become increasingly complex.

The upcoming two-volume set contains all of the previous DiSC materials, with the exception of most IEEE and Springer publications. The proceedings of ER'05 and ER'06 as well as two volumes of The VLDBJ are, however, present, as are some additional conference proceedings (CIDR, ADC, and APCCM). Close on the heels of DiSC'07/08, we will provide DiSC'09 and possibly by the end of the year, DiSC'10. This incredible effort is spearheaded by Curtis Dyreson.

The different volumes of DiSC will be distributed in hardcopy to those with memberships that include this benefit; and content from the volumes, where we have the necessary permissions, will be available online.

Some time ago, a comprehensive effort aimed at re-vitalizing the website was set in motion. Thanks to substantial efforts headed by Jeffrey Xu Yu, we have a new website in terms of content as well as presentation that uses more modern technologies. This new website will serve the SIGMOD membership much more effectively. And we have great plans for continuing to improve the website.

If you have any comments or suggestions for how we can better address the needs of the membership, please send me e-mail ("my three initials"@cs.au.dk) or track me down at SIGMOD/PODS 2010 in Minneapolis.

Sincerely,
Christian S. Jensen
SIGMOD Vice-Chair

# XQJ – XQuery Java API is Completed

Marc Van Cappellen, Zhen Hua Liu, Jim Melton, Maxim Orgiyan

Progress DataDirect
14 Oak Park Drive
Bedford, MA
marc.van.cappellen@datadirect.com

Oracle
500 Oracle Parkway
Redwood Shore, CA
{zhen.liu,jim.melton,maxim.orgiyan}@oracle.com

## ABSTRACT
Just as SQL is a declarative language for querying relational data, XQuery is a declarative language for querying XML. JDBC provides a standard Java API to interact with variety of SQL engines to declaratively access and manipulate data stored in relational data sources. Similarly, XQJ provides a standard Java API to interact with a variety of XQuery engines to declaratively access and manipulate XML data in variety of XML data sources. XQJ, also known as JSR 225, is designed through the Java Community Process (JCP) [20]. The XQJ specification defines a set of Java interfaces and classes that enable a Java program to submit XQuery expressions to an XQuery engine operating on XML data sources and to consume XQuery results. In this article, we discuss the XQJ API's technical details with its similarities and differences from JDBC, the design philosophies and goals for XQJ, the implementation strategies of XQJ in variety of XQuery engines and their operating environments, and the possible future of XQJ.

## 1. INTRODUCTION
Observing the widely successful deployment of JDBC [19] as a standard API for Java applications to plug and play with a variety of SQL engines with different relational backend data sources, we believe the same requirements and use cases exist for XQuery with XML data sources. Furthermore, due to the existence of many XQuery implementations designed for operating in variety of environments managing both persistent and transient XML data, it is self-evident that Java applications should have a standard uniform API to interact with different XQuery engines and their operating environments. The XQJ efforts were started in late 2003, with its initial API draft available in 2004 [1]. Although the final release of XQJ was completed in 2009 as JSR225 [2], the XQJ core API has been stable since XQuery [5] became a W3C recommendation in 2007. The XQJ reference implementation (RI) and technology conformance kit (TCK) for the publication of the API have been stable since 2007. Commercial and open source implementations for the early releases of XQJ have been available since 2005.

The rest of this paper is organized as follows. Section 2 gives a motivating example showing XQJ usage. Section 3 presents key concepts in XQJ by discussing the details of the main interfaces and their conceptual similarities and differences compared to JDBC. Section 4 discusses the XQJ design philosophy. Section 5 discusses the implementation and design choices of XQJ for a variety of XQuery and XML data source environments. Section 6 concludes the article with discussing the possible future of XQJ. Section 7 acknowledges the primary contributors to XQJ.

## 2. MOTIVATING EXAMPLES
Consider Example 1, which represents the typical basic steps of using XQJ in a Java program with the following key concepts.

**Obtaining XQDataSource and XQConnection objects:** *XQDataSource* is an interface from which *XQConnection* interface objects are obtained. The initial *XQDataSource* object can be created through a typical data source instantiation mechanism in Java. For example, an *XQDataSource* object can be obtained via JNDI lookup or Java property file lookup, or can be explicitly created via calling XQJ specific implementation class for *XQDatasource* interface as shown in the example above. The concepts of *XQDataSource* and *XQConnection* are similar to the concepts of *DataSource* and *Connection* in JDBC respectively.

```
XQDataSource ds = null;
XQConnection conn = null;
XQPreparedExpression expr = null;
XQResultSequence result = null;
try
```

```
{
// obtaining XQDataSource instance
ds = (XQDataSource)Class.forName(
                  "com.jsr225.DataSourceImpl").newInstance();
// obtaining connection
 conn = ds.getConnection("usr", "passwd");
// preparing XQuery expression
String xqry = "declare variable $dname as xs:string external;
 for $i in fn:collection('dept')
 where $i/deptname = $dname
 return
 <dinfo>
   <dname>{$dname}</dname>
   <empcnt>{ count($i/employees)}</empcnt>
 </dinfo>";
expr = con,prepareExpression(xqry);
//bind variable with value
expr.bindString(new QName("dname"), "engineering", null);
// execute the XQuery Expression
XQResultSequence rs = expr.executeQuery();
// Consume results
while (rs.next())
{
  Node domNode = rs.getNode();
// do something with the DOM node
}
} catch (XQException e)
{
  e.printStackTrace();
}
finally
{
// clean up resource
  if (rs != null)
 {
  try {rs.close();} catch (XQException e1) {…}
}
  if (expr != null)
 {
  try {expr.close();} catch (XQException e2) {…}
}
  if (conn != null)
 {
  try {conn.close();} catch (XQException e3) {…}
}
```

**Example 1 - XQJ Motivating Example**

**Preparing and Executing an XQuery:** Once an *XQConnection* object is obtained, the XQJ application can execute XQuery using either *XQExpression* or *XQPreparedExpression* interfaces. The difference between the two is that *XQPreparedExpression* is designed to enable users to prepare one XQuery expression and execute it multiple times, each time with possibly different bind values. As shown in Example 1, XQJ applications may pass in a department name as a bind variable so that the same XQuery prepared expression can be re-used to compute different department employee count values with different bind values for the department name. *XQExpression*, on the other hand, is designed to execute an XQuery expression once. That is, a given *XQExpression* object can only be used to evaluate exactly one XQuery expression whereas a given object *XQPreparedExpression* can be reused to execute different XQuery expressions. The concepts of *XQPreparedExpression* and *XQExpression* are similar to the concepts of *PreparedStatement* and *Statement* in JDBC, respectively.

**Consuming an XQuery Result:** Execution of an XQuery results in an XQuery data model (XDM) [6] instance. The *XQResultSequence* interface allows applications to iterate through each item of the result sequence. XQJ applications can obtain each item as needed, which can be either an atomic value or an XML node. This step is similar to the process of iterating through rows in a JDBC *ResultSet*.

**Releasing Resources:** Once the XQuery results have been consumed, XQJ applications release the resources by calling corresponding close methods on *XQResultSequence*, *XQPreparedExpression*, *XQExpression* and *XQConnection* interfaces. Use of Java try/catch constructs to catch *XQException* objects and to ensure proper closing of resources are important to avoid resource leakage.

## 3. XQJ Requirements and Key Concepts

While Section 2 presented aspects of XQJ that are similar to JDBC, there are many key differences between the two. The following is a set of unique requirements for the design of XQJ:

- Providing support for static and dynamic context concepts that are unique to the XQuery language.
- Providing a deferred variable binding mode for binding an XML stream as an external variable input.

- Providing XDM-specific factory methods that allow creation and destruction of XQuery item and sequence objects and XQuery Sequence Type objects. These XDM objects have an independent lifecycle that is unrelated to the lifetime of *XQConnection* and *XQResultSequence* objects.
- Providing mappings for conversion of Java objects to XDM instances in the case of binding XQuery external variables and context item, and XDM instances to Java objects in the case of consumption of XQuery results. In particular, XQJ supports ways of consuming XQuery results using the common XML-related Java interfaces, such as DOM, SAX, and StAX.
- Providing fine-grain exception classes for better diagnosability.

## 3.1 XQuery Static Context & Dynamic Context Interface

XQuery has concepts of static context and dynamic context. XQJ provides *XQStaticContext* and *XQDynamicContext* interfaces to model them. The *XQStaticContext* provides methods that allow applications to get and set various XQuery static context components such as the Base URI, statically known namespaces, and default collation. The *XQStaticContext* object can be obtained by calling *getStaticContext()* on an *XQConnection* object. However, for a single *XQConnection* object, there can be different XQuery expressions prepared and executed, and each of these expression may need to set different values for certain static context components. Therefore, the association of *XQStaticContext* objects with *XQConnection, XQPreparedExpression*, and *XQExpression* objects is passed by value. In other words, a separate copy of the *XQStaticContext* object is made whenever an *XQPreparedExpression* or an *XQExpression* object is created from an *XQConnection* object, so that changes of the static context components in one particular *XQStaticContext* object are isolated from another. Modifications of *XQStaticContext* object retrieved from *XQConnection* object do not affect static context components associated with the *XQConnection* object until the *setStaticContext()* method is invoked on the *XQConnection* object.

The *XQDynamicContext* interface allows XQJ applications to retrieve and set the implicit time zone and bind values for the context item and the external variables of an XQuery. *XQDynamicContext* interface provides *bindString(), bindInt(), bindNode(), etc.* methods to bind XQuery external variable values with a variety of Java built-in primitive types and objects. Both *XQPreparedExpression* and *XQExpression* extend the *XQDynamicContext* interface.

## 3.2 Deferred Variable Binding Mode

To scale with large data size, mature SQL implementations use iterator-based lazy execution models [17] in which the full SQL query result set is not materialized at once but rather produced one row or a set of rows at a time. Applications use an iterator-based interface to obtain the result of such execution. This is reflected in the *ResultSet.next()* fetching method in JDBC. XQuery can be evaluated in the same iterator manner to scale with large data size [15]. This naturally justifies the *XQResultSequence.next()* JDBC-like fetching method. However, for the case of variable binding, the XQJ and JDBC requirements are different. In JDBC, SQL variable binding supports only simple scalar values. There is no concept of binding relational result sets that can be potentially large in size. However, in XQJ, an XQuery variable binding can be an XML document or an XQuery sequence of any size. Furthermore, the XQuery sequence object can be obtained from an XML stream API providing an iterator-like fetch interface. In such a use case, it makes sense for XQJ implementations to defer the binding of the input XML data stream until the XQuery execution time when the input XML data stream is actually consumed. Therefore, besides the default immediate binding mode, a deferred binding mode can be set in the *XQStaticContext*. In the deferred binding mode, the bind value might not be consumed until the variable is actually accessed by the underlying XQuery engine. This enables lazy value consumption and improves XQuery performance and scalability.

## 3.3 XDM Data Factory Support

XQJ needs to model XQuery Data Model [6] concepts. An XDM instance is a sequence of XQuery items. Each item can be an atomic value or an XML node (document, element, attribute, comment, processing instruction, or text). *XQSequence* is the XQJ interface that models XQDM instances. It contains zero or more *XQItem* interface objects. The *XQItem* interface in XQJ represents an XDM item. XDM is used both as input to an XQuery expression and output from the evaluation of that XQuery expression.

The XDM goes hand in hand with the XQuery type system. The *XQSequenceType* and *XQItemType* are two interfaces in XQJ enabling applications to work with the XQuery type system. The *XQSequenceType* interface represents the sequence type defined in XQuery. The *XQItemType* interface represents an item type defined in XQuery. The *XQItemType* interface extends the *XQSequenceType* interface, but restricts its occurrence indicator to be exactly one. The *XQItemType* interface provides methods to obtain information such as the item kind, the base type, the name of the node (if any), the type name of the node (if any), and the XML schema URI associated with the type (if any).

There are two kinds of *XQSequence* and *XQItem* objects in XQJ, and they differ in how they are obtained. The first kind of XDM object is obtained from the result of XQuery execution. Recall that an object of class *XQResultSequence* (the interface extending *XQSequence*) is obtained by invoking the *executeQuery()* method of an *XQExpression* or an *XQPreparedExpression* object. An object of class *XQResultItem* (the interface extending *XQItem*) is obtained by calling the *getItem()* method of an *XQResultSequence* object. The second kind of XQDM object- is obtained by explicit creation via the *XQDataFactory* interface from which the *XQConnection* interface extends. Once created, these objects have lifetimes that are independent of the lifetime of the *XQConnection* object that created them. They remain valid until the *close()* method is called on them. In contrast, the lifetime of *XQResultSequence* and *XQResultItem* objects obtained from XQuery execution depends on the lifetime of the *XQExpression* or *XQPreparedExpression* objects that created them. The lifetime of the *XQExpression* and *XQPreparedExpression* objects, in turn, depends on the lifetime of the *XQConnection* object that created them. Closing an *XQConnection* object implicitly closes all the XDM instances resulting from the execution of XQueries in the context of this *XQConnection* object.

## 3.4 XDM & Java Object Type Conversion

As stated earlier, an XDM item can be either an atomic value or an XML node. Atomic values can be of a variety of XQuery built-in datatypes, such as *xs:decimal, xs:boolean, xs:integer*, all of which have default Java data type mappings defined by XQJ to facilitate conversion between Java built-in type objects and XDM instances of built-in XQuery types.

For binding XML nodes, XQJ provides methods to interact with different Java interfaces that represent XML documents. The *XQDynamicContext* interface provides various bind methods to create XQuery document nodes from the following Java objects: *java.lang.String*, *java.io.Reader*, *java.io.InputStream*, *java.xml.stream.XMLStreamReader*, and *javax.xml.transform.Source*. There is also a *bindNode()* method for binding *org.w3c.dom.Node* DOM nodes.

For consuming XML nodes, *XQItemAccessor* interface (from which both *XQItem* and *XQSequence* extend) provides various “get” methods to convert XDM nodes into the following Java objects: *java.lang.String, java.io.Writer, java.io.OutputStream, org.w3c.dom.Node, javax.xml.stream.XMLStreamReader*, *javax.xml.transform.Result,* and *org.xml.sax.ContentHandler.* In addition, the *XQSequence* interface provides methods to convert an entire XDM sequence into Java objects representing XML nodes.

## 3.5 Exception Handling

XQJ allows the user to distinguish XQuery static or dynamic errors from other non-XQuery related errors through two exception classes: *XQException* and *XQueryException*. The *XQueryException* class gives access to the error code as defined by XQuery, error location information, and various other attributes.

To potentially report multiple errors during the static analyses or dynamic evaluation phase, *XQException* instances are chained and the XQJ application can invoke *getNextException()* method to retrieve all of the exceptions.

## 4. XQJ Design Philosophy

### 4.1 Support for Multiple XQuery Engines Deployment Environments

Since SQL and XQuery share many commonalities (such as a declarative language, amendability for iterator based set at-a-time processing model, support of bind variables, static compilation and dynamic evaluation phases), XQJ reuses those JDBC concepts that are applicable to XQuery as much as possible. However, we recognize that there are various XQuery implementations targeting different operating environments. While it is true that major RDBMS vendors [7,8,9,10] support XQuery for querying XML document content stored in an RDBMS and in XML views over relational data through the SQL/XML standard [4], XQuery is also supported by XML content server vendors to query pure XML content [11, 18]. XQuery is supported in mid-tier servers to provide uniform query language access to query different backend XML data sources, relational sources, and XML messages [16]. XQuery is also supported via standalone libraries to be embeddable indifferent types of applications [12, 13]. Therefore, XQJ has to be an API separate from JDBC. It cannot assume that XQuery and SQL coexist in one environment. Instead, XQJ provides an API to handle the variety of XQuery deployment environments.

### 4.2 Stylistically Consistency with JDBC drivers

XQJ requires establishment of an *XQConnection* before executing an XQuery. This appears unnatural for single-tier collocated XQuery deployment environments in which the

XQuery engine is embedded into the Java application. However, the *XQConnection* interface does not entail a physical network connection object but rather a handle that needs to be created before executing XQuery. Therefore, an *XQConnection* implementation can be either a light-weight handler type object for a single-tier XQuery embedded environment, or can be a heavy-weight network connection object for a multi-tier XQuery server environment. In the latter case, the connection pooling technique can be facilitated via the *PooledXQConnection* interface defined by XQJ.

The *XQSequence* interface ties the XQDM concept and the iterator based access to items within a single *XQSequence* object. Although it is debatable whether modeling the two separately is conceptually clean, here XQJ follows the design of JDBC *ResultSet* that represents both a set of relational rows and its iterator accessor as one object, because both of these tend to be accessed together.

### 4.3 XQDM Interoperability

Unlike relational result column values, which are scalar values mapped to common built-in Java types, XDM item types can be associated with user defined XML schemas. Ensuring a consistent XML schema repository among all tiers in a distributed environment is a complex and expensive task that applications may not be willing to bear. Furthermore, interpreting XML nodes may require the full context of the XML tree of which the node is a part. That is, XQuery types and XDM instances are not *standalone,* but rather context dependent. Thus, interoperability of XDM among XQJ drivers is an issue. Even for the same XQJ implementation in a client/server architecture, the XDM node exchanged between the XQJ client and the XQuery engine on the server can be passed by value or by reference. Passing by reference could be expensive and requires communication of the XML schema information, and full tree node context information among different tiers. Even for the same XQJ driver, the XQuery engine and the XQJ clients may run in different tiers. So the XML schema information and full tree node context information need to be communicated among XQJ tiers. Although there are mechanisms to support such interoperability, it requires proprietary design among XQJ implementations that is beyond the scope of XQJ. Therefore, XQJ only guarantees interoperability of *XQItem* and *XQSequence* instances having built-in XML Schema types, and *XQItemType* and *XQSequenceType* instances of built-in XML Schema types. It is implementation-defined whether an XDM node is passed by value or by reference. However, even for XDM nodes exchanged among different XQJ drivers and XQJ tiers with pass-by-value semantics, the node preserves all of its descendants, with all nodes being untyped.

## 5. XQJ Driver Architecture & Implementation Choices

In this section, we discuss various implementation choices to effectively and efficiently support XQJ drivers.

### 5.1 XQJ Driver Architecture Choices

**Embeddable XQJ driver for collocated XQuery engine**: in this architecture, the XQuery engine and XQJ driver are collocated in the same JVM that is also shared by the Java application that uses the XQJ API, and there is no physical network connection among the XQJ driver, the XQuery engine, and the application. XDM instances can preserve their full tree node context and type context. Passing XDM instances by reference is supported effortlessly.

**Mid-tier based XQJ Driver for pure XML Content Server**: in this architecture, the XQuery engine runs in an XML content server that provides management of and query over XML content. The client applications that access and query the XML content typically run in different tiers than that of the XML content server. The XQJ driver runs on each client tier and communicates with the XML content server using implementation-specific protocols. It is performance-critical for such an XQJ driver to implement *XQConnection* pooling to scale with a large number of clients. The XQJ client and XQuery server may choose a loosely coupled or a tightly coupled architecture, depending on the application requirements. For the tightly coupled choice, the XML content server and its client can share a common XML schema repository, and XML nodes can be passed by reference (much like in the distributed object database architecture). For the loosely coupled choice, the XML content server and all of its clients may not share the same XML schema repository. XML nodes exchanged between tiers are passed by value. The loosely coupled choice typically gives better scalability than the tightly coupled choice.

**Mid-tier based XQJ Driver for SQL/XML enabled RDBMS**: An SQL/XML-enabled RDBMS supports XQuery and XML via new concepts in SQL, such as the XML type, XMLQuery() functions, and XMLTable table function[14] that are defined in the SQL/XML standard[4]. RDBMS servers already provide support for JDBC drivers that run SQL/XML queries. An XQJ driver can be built on top of the JDBC driver, to leverage all the underlying plumbing from JDBC. Submitting an XQuery using the JDBC driver boils down to essentially running the '*SELECT * FROM XMLTABLE(xquery PASSING BY REF COLUMNS '.' XML BY REF)*' SQL/XML query. However, the XML type implementation in the JDBC driver has to be sophisticated enough to support the XDM model with an optional XML schema attached. For a query returning persistently stored XML nodes, passing nodes by reference can be feasibly supported.

**Mid-tier based XQJ driver for data integration**: in this architecture, XQuery is implemented in the mid-tier with the XQJ driver. Its backend data sources can be relational data sources, XML data sources, or other data sources whose data content can be converted into XML or even XML messages, local XML files, *etc*. The XQJ driver can open different kinds of connections to its underlying data sources and push down connection-specific queries to fetch the data content so that the XQuery engine in the XQJ driver can then further filter and assemble the result, based on the original user XQuery.

### 5.2 Facilitate efficient XQuery Evaluation

XML content has different shapes and sizes. While small to medium size XML documents can be processed as DOM objects, it is generally not a scalable solution to process large XML documents, or large number of XML nodes, as an XDM instance from an XQuery result. Thus, the most efficient XQuery implementations leverage iterator-based streaming evaluation strategies as much as possible, to cope with large XDM instances. XQJ recognizes this and provides constructs to facilitate such lazy evaluation strategies. An efficient XQJ driver implementation can consider the following design ideas:

- When retrieving an XDM sequence from XQuery execution, invoke the *getNext()* call of the underlying XQuery engine to consume one item at a time. Certain XQuery engines [15] may even work at sub-XQItem level by making *getNext()* operate at the same level of event as that of an XML stream reader API (StAX) [3]. In this case, implementing the StAX API for XDM nodes can be in sync with the underlying XQuery engine, thus yielding better memory utilization and performance. This streaming principle can be applied to cases when XDM is consumed as other streams as well.
- Support a deferred binding mode when the XDM result of one XQuery needs to be passed in as a bind variable value for another XQuery, especially when the intermediate XDM result is large in size. This support can enable end-to-end streaming evaluation among XQuery engines.

## 6. Conclusion & Future Work

As XQJ finishes its first release, there are multiple concurrent, ongoing XQuery specification efforts, including the XQuery Update Facility, XQuery Full Text, XQuery 1.1, and the XQuery Scripting Extension. The current XQJ specification only supports XQuery 1.0. Since XQuery Full Text expressions, XQuery 1.1 expressions, and XQuery Update Facility transform expressions are all read-only expressions, supporting them using the current XQJ model requires minimal work. However, supporting full power of the XQuery Update Facility in XQJ requires additional infrastructure. Furthermore, it is also debatable whether the XQuery Scripting Extension should be supported via XQJ, because XQuery scripting is an approach of mixing declarative query and imperative procedural manipulation of XML in one language. This contradicts the approach of embedding a declarative query language like XQuery into an imperative programming language like Java.

Nevertheless, the current release of XQJ provides a simple, easy-to-use, and portable API to support the use of XQuery with XML data sources on the Java platform. We believe its impact will be similar to that of JDBC in the years to come.

## 7. ACKNOWLEDGMENTS

We express our thanks to all JSR225 working group members. The following individuals deserve special acknowledgements: Per Bothner, Andrew Eisenberg, Muralidhar Krishnaprasad, Jason Hunter, Jan-Eike Michels, Karuna Muthiah, and Basuki Soeterman.

## 8. REFERENCES

[1] Andrew Eisenberg, Jim Melton: An Early Look at XQuery API for Java (XQJ). SIGMOD Record 33(2): 105-111 (2004)

[2] JSR225: http://www.jcp.org/en/jsr/detail?id=225

[3] JSR173: http://www.jcp.org/en/jsr/detail?id=173

[4] International Organization for Standardization (ISO). Information Technology-Database Language SQL-Part 14: XML-Related Specifications (SQL/XML)

[5] XQuery: http://www.w3.org/TR/xquery/

[6] XDM: http://www.w3.org/TR/xpath-datamodel/

[7] Zhen Hua Liu, Muralidhar Krishnaprasad, Vikas Arora: Native Xquery processing in oracle XMLDB. SIGMOD Conference 2005: 828-833

[8] Zhen Hua Liu, Sivasankaran Chandrasekar, Thomas Baby, Hui J. Chang: Towards a physical XML independent XQuery/SQL/XML engine. PVLDB 1(2): 1356-1367 (2008)

[9] Shankar Pal, Istvan Cseri, Oliver Seeliger, Michael Rys, Gideon Schaller, Wei Yu, Dragan Tomic, Adrian Baras, Brandon Berg, Denis Churin, Eugene Kogan: XQuery Implementation in a Relational Database System. VLDB 2005: 1175-1186

[10] Kevin S. Beyer, Roberta Cochrane, Vanja Josifovski, Jim Kleewein, George Lapis, Guy M. Lohman, Robert Lyle, Fatma Özcan, Hamid Pirahesh, Normen Seemann, Tuong C. Truong, Bert Van der Linden, Brian Vickery, Chun Zhang: System RX: One Part Relational, One Part XML. SIGMOD Conference 2005: 347-358

[11] Mary Holstege: Big, Fast XQuery: Enabling Content Applications. IEEE Data Eng. Bull. 31(4): 41-48 (2008)

[12] Michael Kay: Ten Reasons Why Saxon XQuery is Fast. IEEE Data Eng. Bull. 31(4): 65-74 (2008)

[13] Marc Van Cappellen, Wouter Cordewiner, Carlo Innocenti: Data Aggregation, Heterogeneous Data Sources and Streaming Processing: How Can XQuery Help? IEEE Data Eng. Bull. 31(4): 57-64 (2008)

[14] F Zemke, M. Rys, K. Kulkarni, J. Michels, B. Reinwald, F. Oczan, Zhen. Hua. Liu, I. Davis, K. Hare, "XMLTable" , ISO/IEC JTC1/SC32 WG3:SIA-051 ANSI NCITS H2 2004-039 http://www.wiscorp.com/H2-2004-039-xmltable.pdf

[15] Daniela Florescu, Chris Hillery, Donald Kossmann, Paul Lucas, Fabio Riccardi, Till Westmann, Michael J. Carey, Arvind Sundararajan: The BEA streaming XQuery engine. VLDB J. 13(3): 294-315 (2004)

[16] Michael J. Carey: Data delivery in a service-oriented world: the BEA aquaLogic data services platform. SIGMOD Conference 2006: 695-705

[17] G. Graefe. Query Evaluation Techniques for Large Databases. ACM Computing Surveys, 25(2):73–170, 1993.

[18] EMC-XHIVE http://www.emc.com/domains/x-hive/index.htm

[19] JDBC: http://www.jcp.org/en/jsr/detail?id=221

[20] Java Community Process: http://jcp.org/en/home/index

# The Ubiquitous DBMS

Kyu-Young Whang[1], Il-Yeol Song[2], Taek-Yoon Kim[1], and Ki-Hoon Lee[1]

[1] Department of Computer Science, KAIST, Daejeon, Korea, {kywhang, tykim, khlee }@mozart.kaist.ac.kr

[2] College of Information Science and Technology, Drexel University, Philadelphia, USA, songiy@drexel.edu

## ABSTRACT
Advancement in mobile computing technologies has prompted strong needs for database systems that can be used in small devices such as sensors, cellular phones, PDAs, car navigators, and Ultra Mobile PCs (UMPCs). We term the database systems that are customizable for small computing devices as *Ubiquitous Database Management Systems* (*UDBMSs*). In this paper, we first review the requirements of the UDBMS. The requirements identified include lightweight DBMSs, selective convergence, flash-optimized storage systems, data synchronization, support of unstructured/semi-structured data, complex database operations, self-management, and security. Next, we review existing systems and research prototypes. We review the functionality of UDBMSs including the footprint size, support of standard SQL, transaction management, concurrency control, recovery, indexing, and access control. We then review the supportability of the requirements by those UDBMSs surveyed. We finally present research issues related to the UDBMS.

## 1. INTRODUCTION
The growing popularity of mobile technologies and advancement in computing power have prompted strong needs for database systems that can be used in small computing devices such as sensors, smart cards, cellular phones, PDAs, car navigators, and Ultra Mobile PCs (UMPCs). These small devices with mobility and embedded processors are called *ubiquitous devices* [WH04]. As the ubiquitous devices get computationally powerful and the bandwidth of the wireless network rapidly expands, we can use them to perform tasks anytime and anywhere often downloading a variety of data from servers and uploading sensor data to servers. This kind of computing environment is commonly called the *ubiquitous environment*. New storage devices suitable for ubiquitous devices such as flash memory [GT05] and MEMS (Micro-Electro-Mechanical Systems)-based storage devices [SG04] have been developed. As the capacity of the storage devices is getting bigger and bigger, we can easily store and manage a huge amount of data in a ubiquitous device. This trend prompted strong needs for the database systems that can be used in ubiquitous devices. We term the database systems customizable for small computing devices as *Ubiquitous Database Management Systems* (*UDBMSs*).

Representative *ubiquitous devices* include sensors, smartcards, cellular phones, PDAs, car navigators, and UMPCs. Sensors and smartcards are extremely small in size and have low computing power. A sensor is as small as a coin and is used for gathering data from its surrounding environment and for processing the data. A smartcard embeds a CPU, memory, and storage device for storing and managing data. The other devices are small in size but have high computing power. A cellular phone is used for managing personal information and playing multimedia data. A car navigator is used for finding the shortest way from the current location to the destination. A PDA is used for personal business management and for supporting applications such as an e-mail client, word processor, and spreadsheet. A UMPC is a general purpose PC but much smaller than laptops. Various applications that used to be run on a server can now be run on a UMPC.

The primary *storage* of ubiquitous devices is flash memory. Flash memory is non-volatile and has many advantages over the disk. Since the capacity of flash memory is increasing and the cost decreasing, flash memory will be widely used also in PCs and servers. Another type of storage media for ubiquitous devices is MEMS-based storage devices. A MEMS-based storage device is a secondary storage device and also has many advantages over the disk. Currently, there are some prototypes of MEMS-based storage devices but no products are available yet.

Ubiquitous devices usually have a limited storage capacity. Hence, users store bulk of data in the server and download the necessary parts to the ubiquitous devices. In this environment, when the data are modified in the ubiquitous device, the data need to be transmitted back and stored at the server to maintain consistency of data between them. This issue is called *data synchronization* [IBM06].

As the applications for ubiquitous devices are diverse, many *different types of data* need to be handled. The data types that need to be supported in ubiquitous devices include text data, web pages, XML data, spatial data, multimedia data, and sensor/stream data. E-mail clients, word processors, and spreadsheet applications manage text data. Web browsers manage web pages and XML data. Car navigation systems manage spatial data. Image viewers, MP3's, and movie players manage multimedia data such as JPEG, MP3, and AVI files. A sensor transmits the data sensed to the server as a stream.

To support the ubiquitous environment, the UDBMS needs to be able to be deployed to different types of ubiquitous devices and to be able to support various applications. In addition, the UDBMS should support new types of storage devices, different types of data, data synchronization, self-management, and security.

There are some research prototypes and commercial products of UDBMSs. Representative research prototypes are TinyDB [FHM07], PicoDBMS [ABP07, ABPV08], and Odysseus/Mobile [1]. Representative commercial products are IBM DB2 Everyplace [IBM06, IBM08], Oracle 10g Lite [Orac06], Oracle Berkeley DB [Orac08, SO07], and Microsoft SQL Server CE (Compact Edition) [DS07, MS05]. TinyDB and PicoDBMS have been developed for extremely small devices with low computing power. TinyDB runs on sensors, and PicoDBMS on smartcards. The other DBMSs have been developed for small devices with high computing power such as cellular phones, PDAs, and UMPCs.

Research on UDBMSs is still at an initial stage. Research groups and commercial DBMS vendors have mainly focused on shrinking the footprint size but have not actively dealt with other important problems such as managing various types of data and supporting flash memory and MEMS-based storage devices.

The rest of this paper is organized as follows. In Section 2, we present important requirements of the UDBMS. In Section 3, we survey the functionalities of existing UDBMSs and analyze how well they satisfy the requirements identified. In Section 4, we introduce research issues related to the UDBMS and present the current status and future work of each issue. Finally, in Section 5, we summarize this paper.

## 2. REQUIREMENTS OF THE UDBMS

In this section, we present important requirements of the UDBMS. Nori [Nori07] has presented a broad survey of mobile and embedded DBMSs. Bernhard et al. [BBB+04] have presented open issues and research topics on mobile DBMSs. In this paper, we consider mobile and embedded DBMSs as UDBMSs. Based on existing surveys [BBB+04, Nori07, Whan07], we identify important requirements for the UDBMS as follows.

[1] Odysseus/Mobile is the ubiquitous version of the Odysseus DBMS [WLK+09, WLL+05, WPHL02].

- **Lightweight DBMSs**

Table 1 shows a summary of typical specifications of ubiquitous devices in 2008. As shown in Table 1, ubiquitous devices have lower computing power than PCs or servers.

**Table 1. Typical specifications of ubiquitous devices in 2008.**

| Ubiquitous Devices | CPU Clocks | Main Memory Sizes | Storage Sizes |
|---|---|---|---|
| Sensors | 7 MHz | 0.5 ~ 8 KBytes | 8 ~ 128 KBytes |
| Smartcards | 14 MHz | 4 KBytes | 128 KBytes |
| Cell Phones | 300 MHz | 64 MBytes | 128 MBytes |
| PDAs | 624 MHz | 128 MBytes | 256 MBytes |
| Car Navigators | 1 GHz | 256 MBytes | 16 GBytes |
| UMPCs | 1.3 GHz | 1 GBytes | 80 GBytes |

Using a low-clock CPU, small memory, and small storage, a UDBMS needs to support the functionalities required by applications with acceptable performance. Furthermore, since ubiquitous devices have limited power sources such as batteries, a UDBMS needs to support the functionalities with low power consumption. Thus, it is important to design and implement a UDBMS as simple as possible considering the performance of devices [BBB+04, Nori07]. We also need to consider co-design of hardware and software [ABPV08, BBB+04], specifically, for devices with tight hardware constraints such as sensors and smartcards.

- **Selective Convergence**

To support the lightweight DBMS requirement, it is important to selectively compose the modules of a UDBMS depending on the applications and the device type. In order to emphasize the capacity that selects only necessary modules, we call this property "*selective convergence*" [Whan07]. Selective convergence is a notion that contrasts to extensibility [SAH87] in that it requires all the functionalities be implemented apriori and then customized according to individual needs while extensibility requires new functionalities be easily implemented for added features. For low performance devices such as sensors and smartcards, users would want only simple and basic functionalities. In contrast, for high performance devices such as PDAs and UMPCs, users would want advanced functionalities such as data synchronization. For example, if a user wants to run a GIS application in his PDA, the user would want spatial functionalities. A similar notion has been introduced by Nori [Nori07] as "componentization."

- **New Storage Devices**

For ubiquitous devices, non-volatile memories (e.g., flash, EEPROM, and FeRAM) and very small secondary storage devices (e.g., MEMS) have many advantages over the disk.

Flash memory is a representative non-volatile memory. Compared with the disk, flash memory has attractive features such as small size, better shock resistance, lower power consumption, fast access time, and no mechanical seek and rotational latency [KV08]. Besides, there is an erase operation, which does not exist in the disk. In order to update existing data in a page, an erase operation should be performed first on the entire block to which the page belongs. The erase time is about ten times slower than the write time, and the number of erase operations is limited to 100,000 ~ 1,000,000 times [GT05, NG08].

A MEMS-based storage device is a very small non-volatile secondary storage. The size is as small as 1cm$^2$, and the average access time is ten times faster than that of the disk with lower power consumption. The MEMS device is composed of a media sled and a probe tip array. The *media sled* is a square plate on which data are recorded, and the *probe tip array* is a set of heads. The MEMS device reads and writes data by moving the media sled in the direction of both X and Y axes. By selecting and activating a portion of the heads simultaneously, users can access multiple data sectors in parallel [GSGN00, SG04].

The storage system of the traditional DBMSs has been developed for the disk but not for new storage devices such as flash memory or MEMS devices. To fully exploit the advantages of the new storage devices, we need to develop new storage systems optimized for them [Nori07]. For flash memory, data update time can be optimized by considering overhead caused by updates and the number of erase operations. For MEMS devices, data access time can be optimized by considering data placement and movements of heads.

- **Complex Operations for Advanced Applications**

Advanced database applications require complex operations such as data mining that cannot be implemented by SQL. As ubiquitous devices are rapidly evolving, many advanced applications that have been used in servers will also be required of ubiquitous devices. Thus, a UDBMS should be able to support complex operations.

For example, on PDAs or UMPCs, we foresee the need for running advanced search applications. For example, there may be an application that finds documents similar to a given document. The application needs complex search operations that analyze relationships and similarity between the given document and the stored documents.

In a u-health care environment, we may need advanced applications for patient management. U-health means an anytime and anywhere medical service [IBM09]. In a u-health care environment, sensors are attached to the patient and gather and send the patient's health data to doctors' PDAs. Using PDAs, doctors can check their patients' conditions and decide prescriptions for them anytime and anywhere. As the computing power of the PDA increases, there may be advanced applications that find a patient's walking pattern and abnormal symptoms. The applications need a complex operation that finds meaningful or frequent patterns from a patient's health data.

- **Unstructured and Semi-structured Data**

Various types of unstructured/semi-structured data such as text, multimedia, XML, spatial, stream, and sensor data are widely used in database applications. Those unstructured/semi-structured data are important not only in servers but also in ubiquitous devices. Examples are lyrics data in MP3 players, map data in car navigators, and multimedia data in PDAs. Thus, efficient management and search of unstructured/semi-structured data will also be required of the UDBMS.

- **Data Synchronization**

Ubiquitous devices cannot stay connected to the server all the time due to limited power resources and unstable wireless connection. Furthermore, ubiquitous devices are not able to store a large amount of data due to lack of storage capacity. Thus, users store bulk of data in the server and download only the necessary parts of the data from the server to the ubiquitous device.

In this environment, many users can share and manage data by replicating the same data in the server to their own devices. For example, in a hospital, bulk of patients' data is stored in the server database, and doctors replicate a portion of these data to their PDAs. In this way, many doctors can share and manage the same patient's data anytime and anywhere. As a result, different versions of the data may exist and result in data inconsistency. Therefore, a UDBMS needs to support a functionality to integrate different versions of data into a consistent version [BBB+04, Nori07], i.e., data synchronization.

- **Self-Management**

Unlike in traditional DBMSs, in a UDBMS, there can be no database administrator (DBA) to manage the database. Thus, a UDBMS needs to support self-management functionalities. That is, it needs to automatically perform operations like backup, restore, recovery, indexing, and tuning [Nori07].

- **Security**

Since ubiquitous devices often contain personal data such as banking and healthcare data, a UDBMS needs to ensure the data security by providing access control policies [ABPV08].

- **Longevity and Distributed Processing**

In a wireless sensor network, several thousands of sensor nodes with limited resources, e.g., battery power, are connected through the network to the server (or base station). The server sends queries into the sensor network, and sensor nodes collect, filter, aggregate, and route data back to the server. Thus, a UDBMS in a sensor node needs to support distributed query processing and to minimize power consumption for longevity [MFHH05].

## 3. EXISTING SYSTEMS

In this section, we survey representative research prototypes and commercial products of the UDBMS and compare their functionalities. Then, we review the supportability of the requirements identified in Section 2 for the UDBMSs surveyed.

### 3.1 Representative Systems

Some research groups and commercial DBMS vendors have developed UDBMSs. Table 2 shows the research prototypes and commercial products we surveyed. Commercial products include IBM DB2 Everyplace, Oracle 10g Lite, Oracle Berkeley DB, and Microsoft SQL Server CE. Oracle Berkeley DB has been developed by University of California, Berkeley, but its license has moved to Oracle corp.

**Table 2. Research prototypes and commercial products of the UDBMS.**

| Research Prototypes | Commercial Products |
|---|---|
| TinyDB, PicoDBMS, Odysseus/Mobile | IBM DB2 Everyplace, Oracle 10g Lite, Oracle Berkeley DB, MS SQL Server CE |

Research prototypes include TinyDB, PicoDBMS, and Odysseus/Mobile. TinyDB has been developed at University of California, Berkeley. PicoDBMS has been developed at University of Versailles and INRIA. Odysseus/Mobile is the ubiquitous version of the Odysseus DBMS [WLK+09, WLL+05, WPHL02] that has been continually evolving for the last 19 years at KAIST. Odysseus DBMS is tightly coupled with information retrieval (IR) and spatial database functionalities.

Odysseus/Mobile supports all the functionalities of the Odysseus DBMS and additionally supports selective convergence, data synchronization, and the flash-optimized storage system (ongoing) for ubiquitous devices. Odysseus/Mobile supports selective convergence through the architecture that allows users to choose necessary modules at compile time.

Existing UDBMSs can be categorized by target devices in which the DBMS is deployed. Table 3 shows the summary. In Table 3, it seems that sensors, smartcards, and high performance devices are different enough to justify different DBMSs. However, we expect that the difference will become less obvious as device technology evolves, and UDBMSs that support the requirements in Section 2.1 will be needed for all those devices. For example, even in sensors, complex operations such as data mining will be needed to detect and filter out outliers in sensed data.

**Table 3. Existing UDBMSs categorized by target devices.**

| Target Devices | | UDBMSs |
|---|---|---|
| **Extremely Small Devices with Low Computing Power** | Sensors | TinyDB |
| | Smartcards | PicoDBMS |
| **Small Devices with High Computing Power** | Cell Phones, PDAs, Car Navigators, and UMPCs | IBM DB2 Everyplace, Oracle 10g Lite, Oracle Berkeley DB, MS SQL Server CE, Odysseus/Mobile |

### 3.2 Functional Analysis

Table 4 shows the functionalities of existing UDBMSs. General functionalities of server DBMSs include SQL query processing, views, integrity constraints, transaction management, concurrency control, recovery, indexing, access control, encryption, and compression[2]. Existing UDBMSs support only essential functionalities among those of server DBMSs.

TinyDB supports only essential functionalities for sensor applications. Since most of the sensor applications are used to filter out some data, they just need the functionality that selects data satisfying given conditions. Thus, TinyDB supports only SELECT statements of TinySQL[3], and its footprint is extremely small—only 3 KBytes. For more information on TinyDB, refer to Madden et al. [FHM07].

PicoDBMS supports sufficient functionalities for smartcard applications. Smartcard applications are used for data management such as insert, delete, update and search. Thus, PicoDBMS supports a part of SQL such as INSERT, UPDATE, DELETE, SELECT (with join and aggregation), and CREATE/DROP TABLE statements. Additionally, it supports CREATE/DROP VIEW and GRANT/REVOKE statements. PicoDBMS

[2] Memory consumption is also important, but we excluded since it is not published by commercial vendors or research groups.

[3] TinySQL supports a very limited part of SQL99 such as INSERT, UPDATE, SELECT, and CREATE/DROP TABLE statements.

also supports functionalities for indexing, transaction management, recovery, access control, and compression. The footprint size of PicoDBMS is about 30 KBytes, larger than TinyDB. For more information on PicoDBMS, refer to Bobineau et al. [ABP07, ABPV08].

Oracle Berkeley DB, Oracle 10g Lite, IBM DB2 Everyplace, Microsoft SQL Server CE, and Odysseus/Mobile support most of the functionalities of server DBMSs since their target devices, such as cellular phones, PDAs, car navigators, and UMPCs, have a lot more computing power than sensors and smartcards. These DBMSs commonly support views, transaction management, concurrency control, recovery, and indexing. All the above-mentioned DBMSs, except Oracle Berkeley DB, support a part of the SQL99 standard, which is broader than that of PicoDBMS. Oracle Berkeley DB, however, uses proprietary APIs instead of SQL. The footprint sizes of these DBMSs range from 350 KBytes to 2.5 MBytes.

There is no existing UDBMS that supports all the requirements of the UDBMS. Table 5 shows the supportability of the requirements for each product we reviewed. The UDBMSs that support selective convergence are Oracle Berkeley DB and Odysseus/Mobile. In these UDBMSs, users can choose functionalities such as concurrency control, recovery, and indexing at the compile time. None of the existing UDBMSs support the flash-optimized storage system and complex database operations for advanced applications. The UDBMSs that support self-management are TinyDB, PicoDBMS, and Oracle BerkeleyDB. For Oracle 10g Lite and MS SQL Server CE, support of self-management is not clearly known. All the existing UDBMSs except TinyDB and Oracle BerkeleyDB support security. The UDBMSs that support the data synchronization functionality are Oracle Berkeley DB, Oracle 10g Lite, IBM DB2 manages XML data, and Odysseus/Mobile manages XML, text, and spatial data. In these DBMSs, management of unstructured/semi-structured data can also be selectively converged to the DBMS.

## 4. RESEARCH ISSUES

In this section, we present five research areas that are important for satisfying the requirements of the UDBMS. We present the current status of each research issue and propose future work.

### 4.1 Lightweight DBMSs

Commercial UDBMSs have shrunken the footprint size and memory usage by simplifying the functionalities of server DBMSs, but commercial vendors have not reported the techniques they adopted. The representative research work that deals with lightweight DBMS techniques is PicoDBMS [ABP07, ABPV08]. The paper proposes a new storage system and a new query processor for smartcards, which have a very small storage and memory. The storage system uses a pointer-based storage model where tuples reference their attribute values by means of pointers to preclude any duplicate value. In addition, considering small memory of smartcards, the paper presents techniques for processing queries with a limited memory capacity.

As the ubiquitous environment evolves, many advanced applications that have been used in servers will also be required of ubiquitous devices. However, even high performance devices such as PDAs and UMPCs still lack computing power to run a fully-fledged DBMS that supports as many functionalities as server DBMSs do. Thus, selective convergence will become a very important technique since it enables lightweight DBMSs by selectively composing the modules of a fully-fledged DBMS. We expect research on designing and implementing a new DBMS architecture that supports selective convergence will soon become active.

### 4.2 Storage Systems for New Storage Devices

Research on flash memory has been actively conducted in the field of operating systems (OS) [Alep02, Ban99, Wood01], and recently, in the field of databases [LM07, LMP+08]. The main goal of the research is to optimize the data update cost considering the cost of the erase operation of flash memory. In flash memory, the unit of read/write operations is a page, but the unit of the erase operation is a block, which consists of multiple pages.

In the early days of flash memory, update of an existing page in a block is performed by the method called *in-place update* [GT05]. In the in-place update method, the updated page is overwritten into the original location of the page. The method consists of the following four steps: (1) read all the pages of the block into memory, (2) update the page, (3) erase the block in the storage, and (4) write all the pages in memory to the erased block. This method has an overhead of reading and writing the entire block and needs an expensive erase operation. To solve these problems, the *out-place update* method [GT05] has been proposed. This method writes the updated page into an empty page and

**Table 4. Functionalities of the existing UDBMSs.**

| | TinyDB | PicoDBMS | Oracle Berkeley DB | Oracle 10g Lite | IBM DB2 Everyplace | MS SQL Server CE | Odysseus/ Mobile |
|---|---|---|---|---|---|---|---|
| **Minimum Code Footprint Size** | 3 KBytes | 30 KBytes | 500 KBytes (embedded Linux, storage system only) | 970 KBytes (Windows) | 350 KBytes (embedded Linux) | 2 MBytes (Windows) | 2.5 MBytes (embedded Linux) |
| | | | | | | | 410 KBytes (embedded Linux, storage system only) |
| **SQL** | SELECT only | a part of SQL99 | N | a part of SQL99 | a part of SQL99 | a part of SQL99 | a part of SQL99 |
| **Views** | N | Y | N | Y | Y | Y | Y |
| **Integrity Constraints** | N | N/A | N | Y | Y | Y | N |
| **API** | TinyDB API | JDBC | Berkeley DB API | JDBC, ODBC, ADO.NET, SODA API | DB2 CLI, JDBC, ODBC, ADO.NET API | ADO.NET API | OOSQL CLI, JDBC, ODBC, PHP, Python API |
| **Transaction Management[4]** | N | Y | Y | Y | Y | Y | Y |
| **Concurrency Control** | N | N | Y | Y | Y | Y | Y |
| **Indexing** | N | Y | Y | Y | Y | Y | Y |
| **Access Control** | N | Y | N | Y | Y | Y | Y |
| **Encryption** | N | N/A | Y | Y | Y | Y | N |
| **Compression** | N | Y | N/A | Y | Y | Y | text only |

**Table 5. Supportability of the requirements of the UDBMS.**

| | TinyDB | PicoDBMS | Oracle Berkeley DB | Oracle 10g Lite | IBM DB2 Everyplace | MS SQL Server CE | Odysseus/ Mobile |
|---|---|---|---|---|---|---|---|
| **Lightweight DBMS** | Y | Y | Y | Y | Y | Y | Y |
| **Selective Convergence** | N | N | Y | N | N | N | Y |
| **Flash-optimized Storage System** | N | N | N | N | N | N | under development |
| **Complex Operations** | N | N | N | N | N | N | N |
| **Unstructured/Semi-structured Data** | N | N | XML only | N | N | N | text, spatial, XML |
| **Data Synchronization** | N | N | Y | Y | Y | Y | Y |
| **Self-Management** | Y | Y | Y | N/A | Y | N/A | N |
| **Security** | N | Y | N | Y | Y | Y | Y |

invalidates the original page. The invalidated pages are erased at once in a batch fashion.

There are two active research topics on flash-memory management—the flash translation layer (FTL) and the flash-optimized storage system. FTL is a layer that emulates a disk using flash memory. Using the FTL, most of commercial OS file systems and UDBMSs with a disk-based storage system are able to run on flash memory [Ban99]. The emulation approach, however, has a disadvantage that it is hard to achieve the best performance since the storage system indirectly manages flash memory. The flash-optimized storage system directly manages flash memory without the FTL and is able to achieve the best performance. It has been developed by adapting the data placement method of the log-structured file system to flash memory [GT05]. Representative flash-optimized OS file systems are JFFS [Wood01] and YAFFS [Alep02]. For the flash-based storage system of DBMSs, a prototype called LGeDBMS [KBL+06] and data update methods [KWS10, LM07] have been proposed.

The specificity of considering flash memory in small devices such as smartcards is that they could have extremely small main memory. Recently, there has been research on flash (or EEPROM) storage system for those devices. Yin et al. [YPM09] and Bolchini et al. [BSST03] have addressed the problem of adapting data structures and algorithms to extremely small main memory considering the hardware constraints of flash (or EEPROM) devices.

[4] In general, transaction management includes concurrency control. However, in a single user environment like PicoDBMS, concurrency control is not needed for transaction management.

Research on the MEMS-based storage device is still at an initial stage. Data placement methods considering the movement of headers have been proposed. Similar to flash memory, there are methods proposed to emulate the disk using the MEMS-based storage device [DM03, GSGN00] and those to develop the MEMS-optimized storage system by directly controlling the device [KWKS09, YAA07].

Research on utilizing flash memory and MEMS-based storage devices for DBMSs will become more active. Especially, as the capacity of flash memory per unit cost increases, flash memory is expected to be widely used as storage devices not only for ubiquitous devices but also for PCs and servers. There are many challengeable optimization problems considering the characteristics of flash memory—including optimization of index structures, buffer management, recovery, query processing, and sorting methods. Adaptation of data structures and algorithms to an extremely small main memory environment is also an important issue.

## 4.3 Data Synchronization

In a mobile environment, data synchronization is a very important issue. However, research on data synchronization between the UDBMS and the server has been rare. Representative synchronization modules are the sync server of IBM DB2 Everyplace [IBM06], the mobile server of Oracle 10g Lite [Ora06], and the active sync of Microsoft SQL Server CE [DS07]. Commercial synchronization solutions consist of client databases on ubiquitous devices, a synchronization server, and a server database. The synchronization server controls the consistency of replicated data in the client and server databases.

Main research issues on data synchronization are (1) efficiently maintaining data synchronization between a huge number of ubiquitous devices and the server, (2) resolving conflicts when there are different versions of the same data among ubiquitous devices and the server, (3) recovering from crash and restarting data synchronization when system failure occurs during data synchronization.

## 4.4 Unstructured/Semi-structured Data

Research on managing unstructured/semi-structured data has been active in the context of server DBMSs and will be also important in the context of UDBMSs. Among various types of unstructured/semi-structured data, text, XML, stream, and spatial data have been actively studied. Recently, DB-IR integration [ACR+05, WPHL02] has become a subject of active research.

For server DBMSs, commercial vendors and research groups have developed database systems for managing unstructured/semi-structured data. For XML data, vendors have released extended versions of their DBMSs that support XML data. Research groups have also published on prototypes such as MonetDB/XQuery (CWI) [BGK+06], dbXML (dbXML Group) [dbXML08], and Odysseus/XML (KAIST) [HLL03, LKWL06]. For stream data, many prototypes have been developed. Representative prototypes are TelegraphCQ (University of California, Berkeley) [CCD+03], NiagaraCQ (University of Wisconsin) [CDTW00], and STREAM (Stanford University) [ABB+03]. For text data and spatial data, commercial vendors and research groups have released their products or prototypes. The Odysseus DBMS supports managing text and spatial data in a tightly-coupled fashion [WLK+09, WLL+05, WPHL02]. An early version of the Odysseus DBMS has been used (1997-2000) for the Naver search engine of NHN Co., which is currently the best portal in Korea.

There are two approaches to support unstructured/semi-structured data. One is the loose coupling approach, and the other is the tight coupling approach. Commercial vendors use the loose coupling approach. In the loosely-coupled architecture, the functionality of managing the data is implemented using the DBMS API on top of the DBMS engine. Thus, the loosely-coupled architecture incurs overhead caused by the high-level (typically, SQL-level) API calls between the unstructured/semi-structured data management module and the DBMS engine. In contrast to commercial vendors, the Odysseus DBMS uses the tight coupling approach. In the tightly coupled architecture, the functionality of managing unstructured/semi-structured data is integrated into the DBMS engine [WLK+09, WLL+05, WPHL02]. The tight coupling architecture eliminates the overhead caused by the high-level API calls—obtaining high performance.

Since ubiquitous devices (including PDAs and UMPCs) lack computing power to run a fully-fledged DBMS that supports unstructured/semi-structured data, research on developing a lightweight version of the unstructured/semi-structured data management module considering the specification and performance of the ubiquitous devices needs to be conducted. For example, in a fully-fledged DBMS, one could implement many existing query processing methods and combine them under a cost-based optimization framework so as to select the best query evaluation plan. However, since ubiquitous devices have small memory and storage, it may not be feasible to include many query processing methods and the optimization module in the UDBMS. Thus, we need to study query processing methods that show robust and predictable performance in a lightweight environment where cost-based optimization is not available [MVT05].

### 4.5 Data Mining for Complex Database Operations

Advanced applications perform complex operations such as finding relationships, trends, and meaningful patterns of data. Moreover, target data of the operations are not only relational but also unstructured/semi-structured. Those operations are hard to be implemented with SQL only, but require more advanced techniques such as data mining, which have been a useful tool for supporting complex operations on relational data as well as unstructured/semi-structured data.

In the context of server DBMSs, there has been active research on data mining for various types of unstructured/semi-structured data [HK05]. Examples are text data mining, web page mining, spatial/temporal data mining, stream data mining, multimedia data mining, and bioinformatics data mining. Currently, active research is in progress on adapting traditional mining techniques for relational data to unstructured/semi-structured data [HK05].

Research on data mining will be expanded to the context of the UDBMS. For advanced applications such as advanced search and u-health care mentioned in Section 2, data mining techniques are very useful in implementing complex operations needed for them. For example, in simple filtering applications for the sensor devices, the sensed data are filtered out by checking whether the data are compatible with given arithmetic conditions such as larger-than, less-than, or equal-to; in advanced filtering applications, however, more complex filtering operations such as detecting outliers can be executed by discovering overall trends of the sensed data [SLMJ07]. Considering the specification and performance of ubiquitous devices, research on developing a lightweight version of mining techniques needs to be conducted.

## 5. SUMMARY

In this paper, we have introduced the concept of the UDBMS and related research issues. We have defined the UDBMS as a customizable database system for small computing devices. We have identified important requirements of the UDBMS such as lightweight DBMSs, selective convergence, flash-optimized storage systems, data synchronization, support of unstructured/semi-structured data, complex database operations, self-management, and security. We then have reviewed the functionalities of existing UDBMSs and research prototypes. We have also discussed how well those existing UDBMSs and research prototypes support the requirements identified. We have found that, as of now, there is no UDBMS that supports all the requirements. Especially, flash-optimized storage systems and complex database operations are not supported by any UDBMS. Finally, we have presented research issues related to the UDBMS for each category of the requirements identified.

## 6. ACKNOWLEDGEMENTS

This work was supported by the National Research Lab orogram (No. 2009-0083120) through National Research Foundation (NRF) of Korea funded by Ministry of Education, Science, and Technology [Whan07]. We deeply appreciate anonymous reviewers' incisive comments on the earlier version that made this paper more complete and readable.

## 7. REFERENCES

[ABB+03] Arasu, A. et al., "STREAM: The Stanford Stream Data Manager," *IEEE Data Eng. Bull.*, 26(1), pp. 19-26, 2003.

[ABP07] Anciaux, N., Bouganim, L., and Pucheral, P., "Future Trends in Secure Chip Data Management," *IEEE Data Eng. Bull.*, 30(3), pp. 49-57, 2007.

[ABPV08] Anciaux, N., Bouganim, L., Pucheral, P., and Valduriez, P., "DiSC: Benchmarking Secure Chip DBMS," *IEEE TKDE*, 20(10), pp. 1363-1377, 2008.

[ACR+05] Amer-Yahia, S., Case, P., Rolleke, T., Shanmugasundaram, J., and Weikum, G., "Report on the DB/IR Panel at SIGMOD 2005," *SIGMOD Record*, 34(4), pp. 71-74, 2005.

[Alep02] Aleph One Ltd., "YAFFS: Yet Another Flash Filing System," http://www.yaffs.net, 2002.

[Ban99] Ban, A., Flash File System Optimized for Page-Mode Flash Technologies, US patent 5,937,425, 1999.

[BBB+04] Bernard, G. et al., "Mobile Databases: a Selection of Open Issues and Research Directions," *SIGMOD Record*, 33(2), pp. 78-83, 2004.

[BGK+06] Boncz, P. et al., "MonetDB/XQuery: a Fast XQuery Processor Powered by a Relational Engine," In *SIGMOD*, pp. 479-490, 2006.

[BSST03] Bolchini, C. et al., "Logical and Physical Design Issues for Smart Card Databases," *TOIS*, 21(3), pp. 254-285, 2003.

[CCD+03] Chandrasekaran, S. et al., "TelegraphCQ: Continuous Dataflow Processing for an Uncertain World," In *CIDR*, pp. 269-280, 2003.

[CDTW00] Chen, J. et al. "NiagaraCQ: A Scalable Continuous Query System for Internet Databases," In *SIGMOD*, pp. 379-390, 2000.

[dbXML08] dbXML 2.0, http://www.dbxml.com, 2008.

[DM03] Dramaliev, I. and Madhyastha, T., "Optimizing Probe-Based Storage," In *FAST*, pp. 379-390, 2003.

[DS07] Dhingra, P. and Swanson, T., *Microsoft SQL Server 2005 Compact Edition*, Sams, 2007.

[FHM07] Franklin, M. J., Hellerstein, J. M., and Madden S., "Thinking Big About Tiny Databases," *IEEE*

*Data Eng. Bull.*, 30(3), pp. 37-48, 2007.

[GSGN00] Griffin, J. L., Schlosser, S. W., Ganger, G. R., and Nagle, D. F., “Operating Systems Management of MEMS-Based Storage Device,” In *OSDI*, pp. 227-242, 2000.

[GT05] Gal, E. and Toledo, S., “Algorithms and Data Structures for Flash Memories,” *Computing Surveys*, 37(2), pp. 138-163, 2005.

[HK05] Han, J. and Kimber, M., “Data Mining—On What Kind of Data?,” In Book *Data Mining: Concepts and Techniques*, 2nd ed., Morgan Kaufmann, 2005.

[HLL03] Han, W., Lee, K., and Lee, B., “An XML Storage System for Object-Oriented/Object-Relational DBMSs,” *JOT*, 2(3), pp. 113-126, 2003.

[IBM06] IBM, *DB2 Everyplace Enterprise Edition Release Notes for Version 9.1*, 2006.

[IBM08] IBM Information Center for DB2 Everyplace v9.1, http://publib.boulder.ibm.com/infocenter/db2e/v9r1/index.jsp, 2008.

[IBM09] IBM Ubiquitous Solution, http://www-903.ibm.com/kr/ubiquitous/ucity/health.html, 2009 (in Korean).

[KBL+06] Kim, G., Baek, S., Lee, H., Lee, H., and Joe, M., "LGeDBMS: a Small DBMS for Embedded System with Flash Memory," In *VLDB*, pp. 1255-1258, 2006.

[KV08] Koltsidas, I. and Viglas, S. D., “Flashing up the Storage Layer,” In *VLDB*, pp. 514-525, 2008.

[KWKS09] Kim, Y., Whang, K., Kim, M., and Song, I., “A Logical Model and Data Placement Strategies for MEMS Storage Devices,” *IEICE Trans. on Information and Systems*, E92-D(11), pp. 2218-2234, 2009.

[KWS10] Kim, Y., Whang, K., and Song, I., “Page-Differential Logging: An Efficient and DBMS-independent Approach for Storing Data into Flash Memory,” In *SIGMOD*, 2010 (to appear).

[LKWL06] Lee, K., Kim, S., Whang, E., and Lee, J., "A Practitioner's Approach to Normalizing XQuery Expressions," In *DASFAA*, pp. 437-453, 2006.

[LM07] Lee, S. and Moon, B., “Design of Flash-Based DBMS: An In-Page Logging Approach,” In *SIGMOD*, pp. 55-66, 2007.

[LMP+08] Lee, S. et al., “A Case for Flash Memory SSD in Enterprise Database Applications,” In *SIGMOD*, pp. 1075-1086, 2008.

[MFHH05] Madden, S. R., Franklin, M. J., Hellerstein, J. M., and Hong, W., “TinyDB: an Acquisitional Query Processing System for Sensor Networks”, *TODS*, 30(1), pp. 122-173, 2005.

[MS05] Microsoft, “SQL Server Compact 3.5,” http://www.microsoft.com/sqlserver/2005/en/us/compact.aspx, 2005.

[MVT05] Moro, M. M., Vagena, Z., and Tsotras, V. J., “Tree-Pattern Queries on a Lightweight XML Processor,” In *VLDB*, pp. 205-216, 2005.

[NG08] Nath, S. and Gibbons, P. B., “Online Maintenance of Very Large Random Samples on Flash Storage,” In *VLDB*, pp. 970-983, 2008.

[Nori07] Nori, A. K., “Mobile and Embedded Databases,” *IEEE Data Eng. Bull.*, 30(3), pp. 3-12, Sept. 2007 (also in 2007 *SIGMOD* as a tutorial abstract, June 2007).

[Orac06] Oracle, *Oracle Database Lite 10g Technical White Paper*, 2006.

[Orac08] Oracle Berkeley DB, http://www.oracle.com/technology/products/berkeley-db/index.html, 2008.

[SAH87] Stonebraker, M., Anton, J., and Hirohama, M., “Extendability in POSTGRES,” *IEEE Data Eng. Bull.*, 10(2), pp. 16-23, 1987.

[SG04] Schlosser, S. W. and Ganger, G. R., “MEMS-Based Storage Devices and Standard Disk Interfaces: A Square Peg in a Round Hole?,” In *FAST,* pp. 87-100, 2004.

[SO07] Seltzer, M. and Oracle Corporation, “BerkeleyDB: A Retrospective,” *IEEE Data Eng. Bull.*, 30(3), pp. 21-28, 2007.

[SLMJ07] Sheng, B., Li, Q., Mao, W., and Jin, W., "Outlier Detection in Sensor Networks," In *MobiHoc*, pp. 219-228, 2007.

[WH04] Wu, J. and Hisa, T., "Analysis of E-commerce Innovation and Impact: a Hypercube Model," *Electronic Commerce Research and Applications*, 3(4), pp. 389-404, 2004.

[Whan07] Whang, K., “Development of Customizable/Lightweight DB Engine Technologies for Ubiquitous Small Devices,” *National Research Lab Program*, National Research Foundation (NRF) of Korea, Proposal Apr. 2007, Project July 2007-June 2012.

[WLK+09] Whang, K. et al., “Tightly-Coupled Spatial Database Features in the Odysseus/OpenGIS DBMS for High-Performance,” *GeoInformatica*, to appear, 2009 (on-line version at http://www.springerlink.com/content/m6851246706v6n65).

[WLL+05] Whang, K. et al., “Odysseus: a High-Performance ORDBMS Tightly-Coupled with IR Features,” In *ICDE*, pp. 1104-1105, 2005. This paper received the Best Demonstration Award.

[Wood01] Woodhouse, D., “JFFS: The Journaling Flash File System,” http://sources.redhat.com/jffs2/jffs2.pdf, 2001.

[WPHL02] Whang, K., Park, B., Han, W., and Lee, Y., Inverted Index Storage Structure Using Subindexes and Large Objects for Tight Coupling of Information Retrieval with Database Management Systems, US Patent 6349308, Feb. 2002. (Appl. No. 09/250,487, Feb. 15, 1999)

[YAA07] Yu, H., Agrawal, D., and Abbadi, A. E., “MEMS-Based Storage Architecture for Relational Databases,” *The VLDB Journal*, 16(2), pp. 251–268, 2007.

[YPM09] Yin S., Pucheral, P., and Meng, X., “A Sequential Indexing Scheme for Flash-Based Embedded Systems,” In *EDBT*, pp. 588-599, 2009.

# Relational Processing of RDF Queries: A Survey

Sherif Sakr and Ghazi Al-Naymat
School of Computer Science and Engineering
University of New South Wales, Sydney, Australia
{ssakr,ghazi}@cse.unsw.edu.au

## ABSTRACT
The Resource Description Framework (RDF) is a flexible model for representing information about resources in the web. With the increasing amount of RDF data which is becoming available, efficient and scalable management of RDF data has become a fundamental challenge to achieve the Semantic Web vision. The RDF model has attracted the attention of the database community and many researchers have proposed different solutions to store and query RDF data efficiently. This survey focuses on using relational query processors to store and query RDF data. We provide an overview of the different approaches and classify them according to their storage and query evaluation strategies.

## 1. INTRODUCTION
The goal of the Semantic Web is to provide a common framework for data-sharing across applications, enterprises, and communities. By giving data semantic meaning (through metadata), this framework allows machines to consume, understand, and reason about the structure and purpose of the data. The core of the Semantic Web is built on the Resource Description Framework (RDF) data model [17]. RDF describes a particular resource using a set of RDF statements of the form (subject, predicate, object) triples, also known as (subject, property, value). The *subject* is the resource, the *predicate* is the characteristic being described, and the *object* is the value for that characteristic.

Efficient and scalable management of RDF data is a fundamental challenge at the core of the Semantic Web. Several research efforts have been proposed to address these challenges [1, 2, 6, 13, 16, 28]. Relational database management systems (RDBMSs) have repeatedly shown that they are very efficient, scalable and successful in hosting types of data which have formerly not been anticipated to be stored inside relational databases such as complex objects [27], spatio-temporal data [5] and XML data [11].

This survey focuses on using relational query processors to store and query RDF data. We give an overview of the different approaches and classifies them according to their storage and indexing strategy. The rest of the paper is organized as follows. Section 2 introduces preliminaries of the RDF data model and the W3C standard RDF query language, SPARQL. It also introduce the main alternative relational approaches for storing and querying RDF. Sections 3,4 and 5 provide the details of the different techniques in each of the alternative relational approaches. Finally, Section 6 concludes the paper and provides some suggestions for possible future research directions on the subject.

## 2. RDF-SPARQL PRELIMINARIES
The Resource Description Framework (RDF) is a W3C recommendation that has rapidly gained popularity as a mean of expressing and exchanging semantic metadata, i.e., data that specifies semantic information about data. RDF was originally designed for the representation and processing of metadata about remote information sources and defines a model for describing relationships among resources in terms of uniquely identified attributes and values. The basic building block in RDF is a simple tuple model, (subject, predicate, object), to express different types of knowledge in the form of fact statements. The interpretation of each statement is that subject $S$ has property $P$ with value $O$, where S and P are resource URIs and O is either a URI or a literal value. Thus, any object from one triple can play the role of a subject in another triple which amounts to chaining two labeled edges in a graph-based structure. Thus, RDF allows a form of reification in which any RDF statement itself can be the subject or object of a triple. One of the clear advantage of the RDF data model is its schema-free structure in comparison to the entity-relationship model where the entities, their attributes and relationships to other entities are strictly defined. In RDF, the schema may evolve over the time which fits well with the modern notion of data management, dataspaces, and its *pay-as-you-go* philosophy [14]. Figure 1 illustrates a sample RDF graph.

The SPARQL query language is the official W3C standard for querying and extracting information from RDF graphs [22]. It represents the counterpart to *select-project-join* queries in the relational model. It is based on a powerful graph matching facility, allows binding variables to components in the input RDF graph and supports conjunctions and disjunctions of triple patterns. In addition, operators akin to relational joins, unions, left outer joins, selections, and projections can be combined to build more expressive queries.

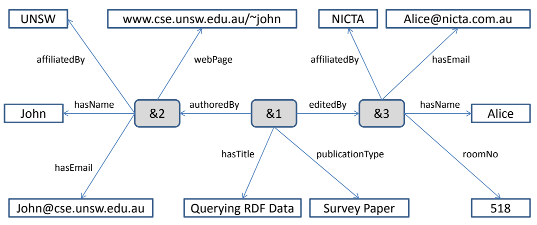
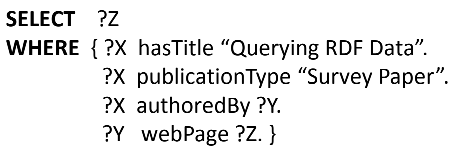


**Figure 1: Sample RDF Graph**



```
SELECT  ?Z
WHERE { ?X  hasTitle "Querying RDF Data".
        ?X  publicationType "Survey Paper".
        ?X  authoredBy ?Y.
        ?Y  webPage ?Z. }
```

**Figure 2: Sample SPARQL query**

A basic SPARQL query has the form:

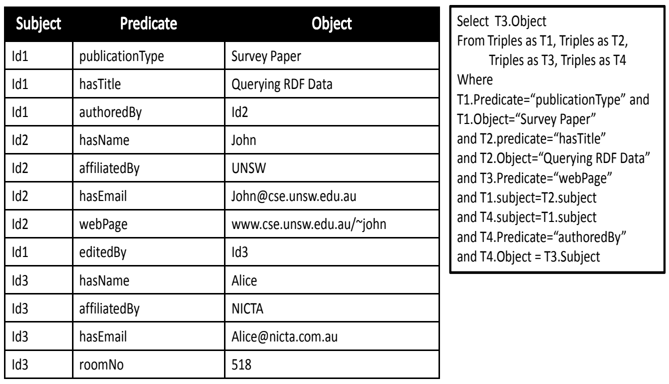
```
select ?variable1 ?variable2 ...
where { pattern1. pattern2. ... }
```

where each pattern consists of *subject*, *predicate* and *object*, and each of these can be either a variable or a literal. The query specifies the known literals and leaves the unknowns as variables which can occur in multiple patterns to constitute join operations. Hence, the query processor needs to find all possible variable bindings that satisfy the given patterns and return the bindings from the projection clause to the application. Figure 2 depicts a sample SPARQL query over the sample RDF graph of Figure 1 to *retrieve the web page information of the author of a book chapter with the title "Querying RDF Data"*.

Relational database management systems (RDBMSs) have repeatedly shown that they are very efficient, scalable and successful in hosting types of data which have formerly not been anticipated to be stored inside relational databases. In addition, RDBMSs have shown their ability to handle vast amounts of data very efficiently using powerful indexing mechanisms. The relational RDF stores can be mainly classified to the following categories:

- **Vertical (triple) table stores:** where each RDF triple is stored directly in a three-column table (subject, predicate, object).
- **Property (n-ary) table stores:** where multiple RDF properties are modeled as n-ary table columns for the same subject.
- **Horizontal (binary) table stores:** where RDF triples are modeled as one horizontal table or into a set of vertically partitioned binary tables (one table for each RDF property).

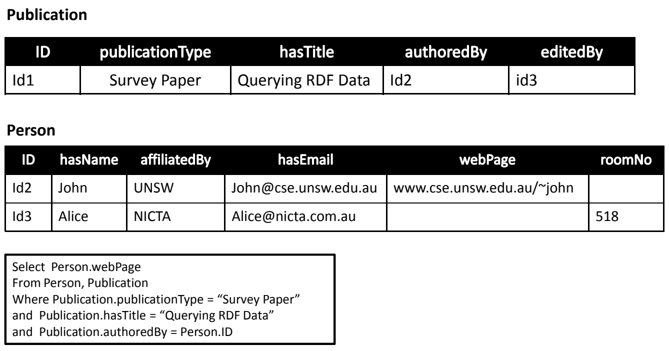
Figures 3,4 and 5 illustrate examples of the three alternative relational representations of the sample RDF graph (Figure 1) and their associated SQL queries for evaluating the sample SPARQL query (Figure 2).



| Subject | Predicate | Object |
|---|---|---|
| Id1 | publicationType | Survey Paper |
| Id1 | hasTitle | Querying RDF Data |
| Id1 | authoredBy | Id2 |
| Id2 | hasName | John |
| Id2 | affiliatedBy | UNSW |
| Id2 | hasEmail | John@cse.unsw.edu.au |
| Id2 | webPage | www.cse.unsw.edu.au/~john |
| Id1 | editedBy | Id3 |
| Id3 | hasName | Alice |
| Id3 | affiliatedBy | NICTA |
| Id3 | hasEmail | Alice@nicta.com.au |
| Id3 | roomNo | 518 |

```
Select  T3.Object
From Triples as T1, Triples as T2,
        Triples as T3, Triples as T4
Where
T1.Predicate="publicationType" and
T1.Object="Survey Paper"
and T2.predicate="hasTitle"
and T2.Object="Querying RDF Data"
and T3.Predicate="webPage"
and T1.subject=T2.subject
and T4.subject=T1.subject
and T4.Predicate="authoredBy"
and T4.Object = T3.Subject
```

**Figure 3: Relational Representation of Triple RDF Stores**



**Publication**

| ID | publicationType | hasTitle | authoredBy | editedBy |
|---|---|---|---|---|
| Id1 | Survey Paper | Querying RDF Data | Id2 | id3 |

**Person**

| ID | hasName | affiliatedBy | hasEmail | webPage | roomNo |
|---|---|---|---|---|---|
| Id2 | John | UNSW | John@cse.unsw.edu.au | www.cse.unsw.edu.au/~john | |
| Id3 | Alice | NICTA | Alice@nicta.com.au | | 518 |

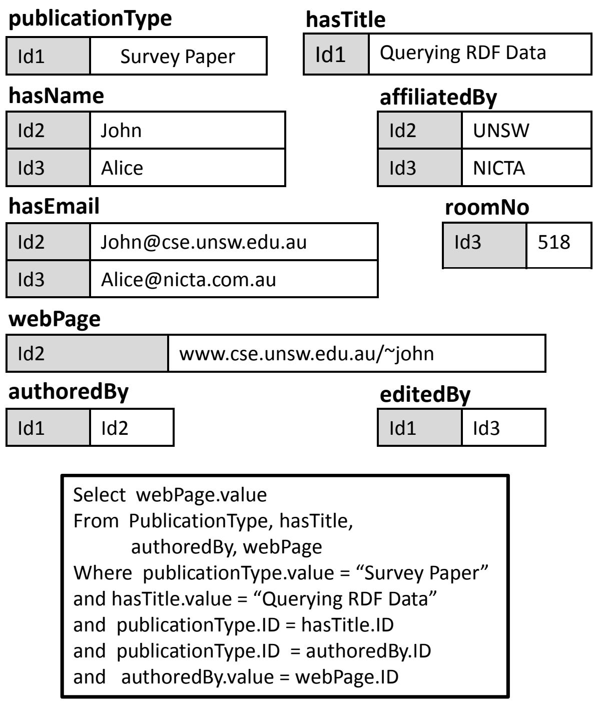
```
Select  Person.webPage
From Person, Publication
Where Publication.publicationType = "Survey Paper"
and  Publication.hasTitle = "Querying RDF Data"
and  Publication.authoredBy = Person.ID
```

**Figure 4: Relational Representation of Property Tables RDF Stores**

## 3. VERTICAL (TRIPLE) STORES

Harris and Gibbins [12] have described the *3store* RDF storage system. The storage system of 3Store is based on a central triple table which holds the hashes for the subject, predicate, object and graph identifier. The graph identifier is equal to zero if the triple resides in the anonymous background graph. A symbols table is used to allow reverse lookups from the hash to the hashed value, for example, to return results. Furthermore it allows SQL operations to be performed on pre-computed values in the data types of the columns without the use of casts. For evaluating SPARQL queries, the triples table is joined once for each triple in the graph pattern where variables are bound to their values when they encounter the slot in which the variable appears. Subsequent occurrences of variables in the graph pattern are used to constrain any appropriate joins with their initial binding. To produce the intermediate results table, the hashes of any SPARQL variables required to be returned in the results set are projected and the hashes from the intermediate results table are joined to the symbols table to provide the textual representation of the results.

Neumann and Weikum [20] have presented the *RDF-3X* (RDF Triple eXpress) RDF query engine which tries to overcome the criticism that triples stores incurs too many expensive self-joins by creating the exhaustive set of indexes and relying on fast processing of merge joins. The physical design of RDF-3x is workload-independent and eliminates the need



**publicationType**

| | |
|---|---|
| Id1 | Survey Paper |

**hasTitle**

| | |
|---|---|
| Id1 | Querying RDF Data |

**hasName**

| | |
|---|---|
| Id2 | John |
| Id3 | Alice |

**affiliatedBy**

| | |
|---|---|
| Id2 | UNSW |
| Id3 | NICTA |

**hasEmail**

| | |
|---|---|
| Id2 | John@cse.unsw.edu.au |
| Id3 | Alice@nicta.com.au |

**roomNo**

| | |
|---|---|
| Id3 | 518 |

**webPage**

| | |
|---|---|
| Id2 | www.cse.unsw.edu.au/~john |

**authoredBy**

| | |
|---|---|
| Id1 | Id2 |

**editedBy**

| | |
|---|---|
| Id1 | Id3 |

```
Select  webPage.value
From  PublicationType, hasTitle,
        authoredBy, webPage
Where  publicationType.value = "Survey Paper"
and hasTitle.value = "Querying RDF Data"
and  publicationType.ID = hasTitle.ID
and  publicationType.ID  = authoredBy.ID
and   authoredBy.value = webPage.ID
```

**Figure 5: Relational Representation of Binary Tables RDF Stores**

for physical-design tuning by building indexes over all 6 permutations of the three dimensions that constitute an RDF triple. Additionally, indexes over count-aggregated variants for all three two-dimensional and all three one-dimensional projections are created. The query processor follows the RISC-style design philosophy [7] by using the full set of indexes on the triple tables to rely mostly on merge joins over sorted index lists. The query optimizer relies upon its cost model in finding the lowest-cost execution plan and mostly focuses on join order and the generation of execution plans. In principle, selectivity estimation has a huge impact on plan generation. While this is a standard problem in database systems, the schema-free nature of RDF data makes the problem more challenging. RDF-3X employs dynamic programming for plan enumeration, with a cost model based on RDF-specific statistical synopses. It relies on two kinds of statistics: 1) specialized histograms which are generic and can handle any kind of triple patterns and joins. The disadvantage of histograms is that it assumes independence between predicates. 2) frequent join paths in the data which give more accurate estimation. During query optimization, the query optimizer uses the join-path selectivity information when available and otherwise assume independence and use the histograms information. In [21] the authors have extended the work further by introducing a run-time technique for accelerating query executions. It uses a light-weight, RDF-specific technique for sideways information passing across different joins and index scans within the query execution plans. They have also enhanced the selectivity estimator of the query optimizer by using very fast index lookups on specifically designed aggregation indexes, rather than relying on the usual kinds of coarse-grained histograms. This provides more accurate estimates at compile-time, at a fairly small cost that is easily amortized by providing better directives for the join-order optimization.

Weiss, et al. [28] have presented the *Hexastore* RDF storage scheme with main focuses on scalability and generality in its data storage, processing and representation. Hexastore is based on the idea of indexing the RDF data in a multiple indexing scheme [13]. It does not discriminate against any RDF element and treats subjects, properties and objects equally. Each RDF element type have its special index structures built around it. Moreover, every possible ordering of the importance or precedence of the three elements in an indexing scheme is materialized. Each index structure in a Hexastore centers around one RDF element and defines a prioritization between the other two elements. Two vectors are associated with each RDF element (e.g. subject), one for each of the other two RDF elements (e.g. property and object). In addition, lists of the third RDF element are appended to the elements in these vectors. In total, six distinct indices are used for indexing the RDF data. These indices materialize all possible orders of precedence of the three RDF elements. A clear disadvantage of this approach is that Hexastore features a worst-case five-fold storage increase in comparison to a conventional triples table.

## 4. PROPERTY TABLE STORES

Due to the proliferations of self-joins involved with the triple-store, the property table approach was proposed. *Jena* is a an open-source toolkit for Semantic Web programmers [19]. It implements persistence for RDF graphs using an SQL database through a JDBC connection. The schema of the first version of Jena, Jena1, consisted of a statement table, a literals table and a resources table. The statement table *(Subject, Predicate, ObjectURI, ObjectLiteral)* contained all statements and referenced the resources and literals tables for subjects, predicates and objects. To distinguish literal objects from resource URIs, two columns were used. The literals table contained all literal values and the resources table contained all resource URIs in the graph. However, every query operation required multiple joins between the statement table and the literals table or the resources table.

To address this problem, the *Jena2* schema trades-off space for time. It uses a denormalized schema in which resource URIs and simple literal values are stored directly in the statement table. In order to distinguish database references from literals and URIs, column values are encoded with a prefix that indicates the type of the value. A separate literals table is only used to store literal values whose length exceeds a threshold, such as blobs. Similarly, a separate resources table is used to store long URIs. By storing values directly in the statement table it is possible to perform many queries without a join. However, a denormalized schema uses more database space because the same value (literal or URI) is stored repeatedly. The increase in database space consumption is addressed by using string compression schemes. Both Jena1 and Jena2 permit multiple graphs to be stored in a single database instance. In Jena1, all graphs were stored in a single statement. However, Jena2 supports the use of multiple statement tables in a single database so that applications can flexibly map graphs to different tables. In this way, graphs that are often accessed together may be stored together while graphs that are never accessed together may be stored separately.

In principle, applications typically have access patterns in which certain subjects and/or properties are accessed together. For example, a graph of data about persons might have many occurrences of objects with properties name, address, phone, gender that are referenced together. Jena2 uses property table as a general facility for clustering properties that are commonly accessed together. A property table is a separate table that stores the subject-value pairs related by a particular property. A property table stores all instances of the property in the graph where that property does not appear in any other table used for the graph. In Jena1, each query is evaluated with a single SQL select query over the statement table. In Jena2, queries have to be generalized because there can be multiple statement tables for a graph. Using the knowledge of the frequent access patterns to construct the property-tables and influence the underlying database storage structures can provide a performance benefit and reduce the number of join operations during the query evaluation process.

Chong et al. [8] have introduced an Oracle-based SQL table function *RDFMATCH* to query RDF data. The results of RDFMATCH table function can be further processed by SQL's rich querying capabilities and seamlessly combined with queries on traditional relational data. The core implementation of RDFMATCH query translates to a self-join query on triple-based RDF table store. The resulting query is executed efficiently by making use of B-tree indexes as well as creating materialized join views for specialized subject-property. Subject-Property Matrix materialized join views are used to minimize the query processing overheads that are inherent in the canonical triple-based representation of RDF. The materialized join views are incrementally maintained based on user demand and query workloads. A special module is provided to analyze the table of RDF triples and estimate the size of various materialized views, based on which a user can define a subset of materialized views. For a group of subjects, the system defines a set of single-valued properties that occur together. These can be direct properties of these subjects or nested properties. A property $p_1$ is a direct property of subject $x_1$ if there is a triple $(x_1, p_1, x_2)$. A property $p_m$ is a nested property of subject $x_1$ if there is a set of triples such as, $(x_1, p_1, x_2), ..., (x_m, p_m, x_{m+1})$, where $m > 1$. For example, if there is a set of triples, $(John, address, addr1), (addr1, zip, 03062)$, then the *zip* property is considered as a nested property of *John*.

Levandoski and Mokbel [15] have presented another property table approach for storing RDF data without any assumption about the query workload statistics. The main goals of this approach are: (1) reducing the number of join operations which are required during the RDF query evaluation process by storing related RDF properties together (2) reducing the need to process extra data by tuning null storage to fall below a given threshold. The approach provides a *tailored* schema for each RDF data set which represents a balance between property tables and binary tables and is based on two main parameters: 1) *Support threshold* which represents a value to measure the strength of correlation between properties in the RDF data. 2) The *null threshold* which represents the percentage of null storage tolerated for each table in the schema. The approach involves two phases: *clustering* and *partitioning*. The clustering phase scans the RDF data to automatically discover groups of related properties (i.e., properties that always exist together for a large number of subjects). Based on the support threshold, each set of $n$ properties which are grouped together in the same cluster are good candidates to constitute a single n-ary table and the properties which are not grouped in any cluster are good candidates for storage in binary tables. The partitioning phase goes over the formed clusters and balances the tradeoff between storing as many RDF properties in clusters as possible while keeping null storage to a minimum based on the null threshold. One of the main concerns of the partitioning phase is twofold. The first is to ensure that there is no overlap between the clusters and that each property exists in a single cluster. The second is to reduce the number of table accesses and unions necessary in query processing.

Matono, et al. [18] have proposed a path-based relational RDF database. The main focus of this approach is to improve the performance for path queries by extracting all reachable path expressions for each resource and store them. Thus, there is no need to perform join operations unlike the flat tripe stores or the property tables approach. In this approach, the RDF graph is divided into subgraphs and then each subgraph is stored by applicable techniques into distinct relational tables. More precisely, all classes and properties are extracted from RDF schema data, and all resources are also extracted from RDF data. Each extracted item is assigned an identifier and a path expression and stored in corresponding relational table.

## 5. HORIZONTAL STORES

Abadi, et al. [1] have presented *SW-Store* as a new DBMS which stores RDF data using a fully decomposed storage model (DSM) [10]. In this approach, the triples table is rewritten into $n$ two-column tables where $n$ is the number of unique properties in the data. In each of these tables, the first column contains the subjects that define that property and the second column contains the object values for those subjects while the subjects that do not define a particular property are simply omitted from the table for that property. Each table is sorted by subject, so that particular subjects can be located quickly, and that fast merge joins can be used to reconstruct information about multiple properties for subsets of subjects. For a multi-valued attribute, each distinct value is listed in a successive row in the table for that property. One advantage of this approach is that while property tables need to be carefully constructed so that they are wide enough but not too wide to independently answer queries, the algorithm for creating tables in the vertically partitioned approach is straightforward and need not change over time. Moreover, in the property-class schema approach, queries that do not restrict on class tend to have many union clauses while in the vertically partitioned approach, all data for a particular property is located in the same table and thus union clauses in queries are less common. The implementation of SW-Store relies on a column-oriented DBMS, C-store [26], to store tables as collections of columns rather than as collections of rows. In standard row-oriented databases (e.g., Oracle, DB2, SQLServer, Postgres, etc.) entire tuples are stored consecutively. The problem with this is that if only a few attributes are accessed per query, entire rows need to be read into memory from disk before the projection can occur. By storing data in columns rather than rows, the projection occurs for free only where those columns that are relevant to a query need to be read.

[3, 9] have argued that storing a sparse data set (like RDF) in multiple tables can cause problems. They suggested storing a sparse data set in a single table while the complexities of sparse data management can be handled inside an RDBMS with the addition of an interpreted storage format. The proposed format starts with a header which contains fields such as relation-id, tuple-id, and a tuple length. When a tuple has a value for an attribute, the attribute identifier, a length field (if the type is of variable length), and the value appear in the tuple. The attribute identifier is the *id* of the attribute in the system catalog while the attributes that appear in the system catalog but not in the tuple are null for that tuple. Since the interpreted format stores nothing for null attributes, sparse data sets in a horizontal schema can in general be stored much more compactly in the format. While the interpreted format has storage benefits for sparse data, retrieving the values from attributes in tuples is more complex. In fact, the format is called interpreted because the storage system must discover the attributes and values of a tuple at tuple-access time, rather than using precompiled position information from a catalog, as the positional format allows. To tackle this problem, a new operator (called *EXTRACT* operator) is introduced to the query plans to precede any reference to attributes stored in the interpreted format and returns the offsets to the referenced interpreted attribute values which is then used to retrieve the values. Value extraction from an interpreted record is a potentially expensive operation that is dependent on the number attributes stored in a row or the length of the tuple. Moreover, if a query evaluation plan fetches each attribute individually and uses an EXTRACT call per attribute, the record will be scanned for each attribute and will thus be very slow. Thus, a batch EXTRACT technique is used to allow for a single scan of the present values in order to save time.

## 6. CONCLUDING REMARKS

RDF is a main foundation for processing the semantic of information stored on the Web. It is the data model behind the Semantic Web vision whose goal is to enable integration and sharing of data across different applications and organizations. The naive way to store a set of RDF statements is using a relational database with a single table including columns for subject, property and object. While simple, this schema quickly hits scalability limitations. Therefore, several approaches have been proposed to deal with this limitation by using extensive set of indexes or by using selectivity estimations to optimize the join ordering [20, 28].

Another approach to reduce the self-join problem is to create separate tables (property tables) for subjects that tend to have common properties defined [8, 15]. Since Semantic Web data is often semi-structured, storing this data in a row-store can result in very sparse tables as more subjects or properties are added. Hence, this normalization technique is typically limited to resources that contain a similar set of properties and many small tables are usually created. The problem is that this may result in union and join clauses in queries since information about a particular subject may be located in many different property tables. This may complicate the plan generator and query optimizer and can degrade performance.

Abadi, et al. [1] have explored the trade-off between triple-based stores and binary tables-based stores of RDF data. The main advantages of binary tables are:

- **Improved bandwidth utilization:** In a column store, only those attributes that are accessed by a query need to be read off disk. In a row-store, surrounding attributes also need to be read since an attribute is generally smaller than the smallest granularity in which data can be accessed.
- **Improved data compression:** Storing data from the same attribute domain together increases locality and thus data compression ratio. Hence, bandwidth requirements are further reduced when transferring compressed data.

On the other side, binary tables do have the following main disadvantages:

- **Increased cost of inserts:** Column-stores perform poorly for insert queries since multiple distinct locations on disk have to be updated for each inserted tuple (one for each attribute).
- **Increased tuple reconstruction costs:** In order for column-stores to offer a standards-compliant relational database interface (e.g., ODBC, JDBC, etc.), they must at some point in a query plan stitch values from multiple columns together into a row-store style tuple to be output from the database.

Abadi et al. [1] have reported that the performance of binary tables is superior to clustered property table while Sidirourgos et al. [25] reported that even in column-store database, the performance of binary tables is not always better than clustered property table and depends on the characteristics of the data set. Moreover, the experiments of [1] reported that storing RDF data in column-store database is better than that of row-store database while [25] experiments have shown that the gain of performance in column-store database depends on the number of predicates in a data set. Other independent benchmarking projects [4, 23, 24] have shown that no approach is dominant for all queries and none of these approaches can compete with a purely relational model. Therefore, they are convinced that there is still room for optimization in the proposed generic relational RDF storage schemes and thus new techniques for storing and querying RDF data are still required to bring forward the Semantic Web vision.

## 7. REFERENCES

[1] Daniel J. Abadi, Adam Marcus, Samuel Madden, and Kate Hollenbach. SW-Store: a vertically partitioned DBMS for Semantic Web data management. *VLDB Journal*, 18(2):385–406, 2009.

[2] Sofia Alexaki, Vassilis Christophides, Gregory Karvounarakis, Dimitris Plexousakis, and Karsten Tolle. The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases. In *Proceedings of the 2nd InternationalWorkshop on the Semantic Web*

*(SemWeb)*, 2001.

[3] Jennifer L. Beckmann, Alan Halverson, Rajasekar Krishnamurthy, and Jeffrey F. Naughton. Extending RDBMSs To Support Sparse Datasets Using An Interpreted Attribute Storage Format. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE)*, page 58, 2006.

[4] Christian Bizer and Andreas Schultz. Benchmarking the Performance of Storage Systems that expose SPARQL Endpoints. In *Proceedings of the 4th International Workshop on Scalable Semantic Web knowledge Base Systems (SSWS).*, 2008.

[5] Viorica Botea, Daniel Mallett, Mario A. Nascimento, and Jörg Sander. PIST: An Efficient and Practical Indexing Technique for Historical Spatio-Temporal Point Data. *GeoInformatica*, 12(2):143–168, 2008.

[6] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In *Proceedings of the First International Semantic Web Conference (ISWC)*, pages 54–68, 2002.

[7] Surajit Chaudhuri and Gerhard Weikum. Rethinking Database System Architecture: Towards a Self-Tuning RISC-Style Database System. In *Proceedings of 26th International Conference on Very Large Data Bases (VLDB)*, pages 1–10, 2000.

[8] Eugene Inseok Chong, Souripriya Das, George Eadon, and Jagannathan Srinivasan. An Efficient SQL-based RDF Querying Scheme. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, pages 1216–1227, 2005.

[9] Eric Chu, Jennifer L. Beckmann, and Jeffrey F. Naughton. The case for a wide-table approach to manage sparse relational data sets. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 821–832, 2007.

[10] George P. Copeland and Setrag Khoshafian. A Decomposition Storage Model. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 268–279, 1985.

[11] Torsten Grust, Sherif Sakr, and Jens Teubner. XQuery on SQL Hosts. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB)*, pages 252–263, 2004.

[12] Stephen Harris and Nicholas Gibbins. 3store: Efficient Bulk RDF Storage. In *Proceedings of the First International Workshop on Practical and Scalable Semantic Systems (PSSS)*, 2003.

[13] Andreas Harth and Stefan Decker. Optimized Index Structures for Querying RDF from the Web. In *Proceedings of the Third Latin American Web Congress (LA-WEB)*, pages 71–80, 2005.

[14] Shawn R. Jeffery, Michael J. Franklin, and Alon Y. Halevy. Pay-as-you-go user feedback for dataspace systems. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 847–860, 2008.

[15] Justin J. Levandoski and Mohamed F. Mokbel. RDF Data-Centric Storage. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*, 2009.

[16] Li Ma, Zhong Su, Yue Pan, Li Zhang, and Tao Liu. RStar: an RDF storage and query system for enterprise resource management. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 484–491, 2004.

[17] Frank Manola and Eric Miller. RDF Primer, W3C Recommendation, February 2004. http://www.w3.org/TR/REC-rdf-syntax/.

[18] Akiyoshi Matono, Toshiyuki Amagasa, Masatoshi Yoshikawa, and Shunsuke Uemura. A Path-based Relational RDF Database. In *Proceedings of the 16th Australasian Database Conference (ADC)*, pages 95–103, 2005.

[19] Brian McBride. Jena: A Semantic Web Toolkit. *IEEE Internet Computing*, 6(6):55–59, 2002.

[20] Thomas Neumann and Gerhard Weikum. RDF-3X: a RISC-style engine for RDF. *Proceedings of the VLDB Endownment (PVLDB)*, 1(1):647–659, 2008.

[21] Thomas Neumann and Gerhard Weikum. Scalable join processing on very large RDF graphs. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 627–640, 2009.

[22] Eric Prud'hommeaux and Andy Seaborne. SPARQL Query Language for RDF, W3C Recommendation, January 2008. http://www.w3.org/TR/rdf-sparql-query/.

[23] Michael Schmidt, Thomas Hornung, Norbert Küchlin, Georg Lausen, and Christoph Pinkel. An Experimental Comparison of RDF Data Management Approaches in a SPARQL Benchmark Scenario. In *Proceedings of the 7th International Semantic Web Conference (ISWC)*, pages 82–97, 2008.

[24] Michael Schmidt, Thomas Hornung, Georg Lausen, and Christoph Pinkel. SP2Bench: A SPARQL Performance Benchmark. In *Proceedings of the 25th International Conference on Data Engineering (ICDE)*, pages 222–233, 2009.

[25] Lefteris Sidirourgos, Romulo Goncalves, Martin L. Kersten, Niels Nes, and Stefan Manegold. Column-store support for RDF data management: not all swans are white. *Proceedings of the VLDB Endownment (PVLDB)*, 1(2):1553–1563, 2008.

[26] Michael Stonebraker, Daniel J. Abadi, Adam Batkin, Xuedong Chen, Mitch Cherniack, Miguel Ferreira, Edmond Lau, Amerson Lin, Samuel Madden, Elizabeth J. O'Neil, Patrick E. O'Neil, Alex Rasin, Nga Tran, and Stanley B. Zdonik. C-Store: A Column-oriented DBMS. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, pages 553–564, 2005.

[27] Can Türker and Michael Gertz. Semantic integrity support in SQL: 1999 and commercial (object-)relational database management systems. *VLDB Journal*, 10(4):241–269, 2001.

[28] Cathrin Weiss, Panagiotis Karras, and Abraham Bernstein. Hexastore: sextuple indexing for semantic web data management. *Proceedings of the VLDB Endownment (PVLDB)*, 1(1):1008–1019, 2008.

# Social Sites Research Through CourseRank

Benjamin Bercovitz, Filip Kaliszan, Georgia Koutrika,
Henry Liou, Aditya Parameswaran, Petros Venetis,
Zahra Mohammadi Zadeh, Hector Garcia-Molina
Computer Science Department, Stanford University, California, USA
{berco, kaliszan, koutrika, liouh, adityagp, venetis, zahram, hector}@stanford.edu

## ABSTRACT

Social sites such as FaceBook, Orkut, Flickr, MySpace and many others have become immensely popular. At these sites, users share their resources (e.g., photos, profiles, blogs) and learn from each other. On the other hand, higher education applications help students and administrators track and manage academic information such as grades, course evaluations and enrollments. Despite the importance of both these areas, there is relatively little research on the mechanisms that make them effective. Apart from being both a successful social site and an academic planning site, CourseRank provides a live testbed for studying fundamental questions related to social networking, academic planning, and the fusion of these areas. In this paper, we provide a system overview and our main research efforts through CourseRank.

## 1. INTRODUCTION

A growing number of social sites can be found on the Web enabling people to share different kinds of resources, such as: photos (e.g., Flickr [4]), URLs (e.g., Del.icio.us [3]), blogs (e.g., Technorati [17]), and so forth. These sites differ from the open Web in that they tend to foster communities of registered users that contribute regularly and are controlled by some entity that can set up "rules" of engagement.

The increasing popularity of these systems has motivated a number of studies (e.g., [2, 5, 6]) that have mainly focused on understanding the usage and evolution of these systems as well as a number of efforts on harvesting social knowledge for tasks, such as resource recommendations [13, 14, 18], expert and community identification [10, 19] and ontology induction [16]. Still, there are many unanswered questions about how people interact and what services should and can be offered in social sites. During the summer of 2007, we decided that if we wanted to investigate social sites, we needed to have our own site. The result of our effort is CourseRank, a social site where Stanford students can review courses and plan their academic program.

By focusing on an academic site, not only can we study social sharing and networking, but we can also study an important application area for database systems that has seen little research: *higher-education applications*. There are over 6000 Universities in the USA alone, with over 15M college students, most of whom use software to track courses that they take. Several companies, including Red Lantern (DARS), Jenzabar, Datatel, and PeopleSoft (Oracle), have products for course planning (but not centered on a student community). Thus, our work "kills two birds with one stone", investigating not just social sites, but an interesting and important application beyond "sharing videos and chatting about movies" as in many current social sites.

In the next section we describe the current CourseRank system and what sets it apart from other social sites. Then, in Section 3, we give an overview of our research work in CourseRank. In Section 4, we discuss evaluating social aspects of the system.

## 2. COURSERANK SYSTEM DESCRIPTION

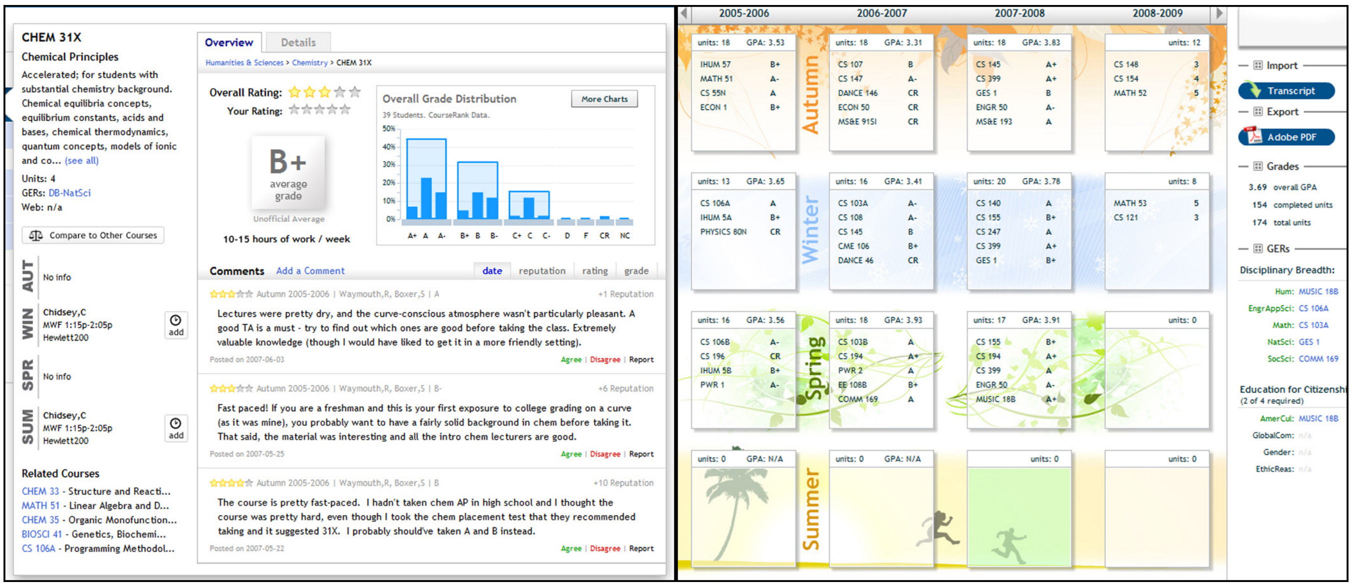
Using CourseRank, students can search for courses of interest, evaluate courses taken and receive personalized recommendations based on what other students have taken and liked. Faculty can also add comments to their own courses, and can see how their class compares to other classes. CourseRank has been successful (it is already used by more than 10,000 students at Stanford, out of a total of about 14,000 students) because in addition to features common to other social sites, it provides special tools geared to the academic domain. For instance, Figure 1 shows two CourseRank screen shots: on the left is part of a course description page, while on the right is the 4-year course planner that helps students structure their courses over multiple years.

In addition, unlike other social sites, it provides access to three types of information:

- *Official Stanford Data.* We have access to official information, including course descriptions and schedules, and results of the official course evaluations conducted by the university.
- *Personal Information.* Students provide personal information (e.g., their class, major), the courses they have already taken and their grades.
- *Evaluations.* Students can evaluate courses they have taken and enter comments. This component is similar to what commercial sites offer, e.g., for rating and reviewing books at Amazon.com.

## 3. DATA-CENTERED SERVICES

In CourseRank, unlike other public course evaluation sites (e.g., RateMyProfessors.com) and social sites, we have access to much richer data: In addition to basic information (what courses a student took and which ones he or she liked), we have other types of information: what grade the student received, how the courses interrelate (needed for major, pre-requisites), and user profiles (major, class, ...). This rich data gives us the opportunity to develop novel types of data-centered services, where the user interacts mainly with the data that the system has collected from all users. These services include: a recommendation engine that lets students personalize their recommendations; a requirements service that checks if students have met program requirements, and incorporates missing courses into recommendations; a novel interface that facilitates discovery of new courses. For each service we explore models for representing information (e.g., course requirements), algorithms and options for implementing them, desirable user interfaces, expected performance, and social implications.



CHEM 31X

Chemical Principles

Accelerated; for students with substantial chemistry background. Chemical equilibria concepts, equilibrium constants, acids and bases, chemical thermodynamics, quantum concepts, models of ionic and co... (see all)

Units: 4
GERs: DB-NatSci
Web: n/a

Compare to Other Courses

AUT No info

WIN Chidsey,C MWF 1:15p-2:05p Hewlett200 add

SPR No info

SUM Chidsey,C MWF 1:15p-2:05p Hewlett200 add

Related Courses

CHEM 33 - Structure and Reacti...
MATH 51 - Linear Algebra and D...
CHEM 35 - Organic Monofunction...
BIOSCI 41 - Genetics, Biochemi...
CS 106A - Programming Methodol...

Overview | Details

Humanities & Sciences > Chemistry > CHEM 31X

Overall Rating: ★★★☆☆
Your Rating: ☆☆☆☆☆

B+ average grade
Unofficial Average

10-15 hours of work / week

Overall Grade Distribution
39 Students. CourseRank Data.

Comments Add a Comment | date | reputation | rating | grade

★★★☆☆ Autumn 2005-2006 | Waymouth,R, Boxer,S | A +1 Reputation
Lectures were pretty dry, and the curve-conscious atmosphere wasn't particularly pleasant. A good TA is a must - try to find out which ones are good before taking the class. Extremely valuable knowledge (though I would have liked to get it in a more friendly setting).
Posted on 2007-06-03 Agree | Disagree | Report

★★★☆☆ Autumn 2005-2006 | Waymouth,R, Boxer,S | B- +6 Reputation
Fast paced! If you are a freshman and this is your first exposure to college grading on a curve (as it was mine), you probably want to have a fairly solid background in chem before taking it. That said, the material was interesting and all the intro chem lecturers are good.
Posted on 2007-05-25 Agree | Disagree | Report

★★★★☆ Autumn 2005-2006 | Waymouth,R, Boxer,S | B +10 Reputation
The course is pretty fast-paced. I hadn't taken chem AP in high school and I thought the course was pretty hard, even though I took the chem placement test that they recommended taking and it suggested 31X. I probably should've taken A and B instead.
Posted on 2007-05-22 Agree | Disagree | Report

| | 2005-2006 | 2006-2007 | 2007-2008 | 2008-2009 |
|---|---|---|---|---|
| Autumn | units: 18 GPA: 3.53<br>IHUM 57 B+<br>MATH 51 A-<br>CS 55N A<br>ECON 1 B+ | units: 18 GPA: 3.31<br>CS 107 B<br>CS 147 A-<br>DANCE 146 CR<br>ECON 50 CR<br>MS&E 91SI CR | units: 18 GPA: 3.83<br>CS 145 A+<br>CS 399 A+<br>GES 1 B<br>ENGR 50 A-<br>MS&E 193 A | units: 12<br>CS 148 3<br>CS 154 4<br>MATH 52 5 |
| Winter | units: 13 GPA: 3.65<br>CS 106A A<br>IHUM 5A B+<br>PHYSICS 80N CR | units: 16 GPA: 3.41<br>CS 103A A-<br>CS 108 A-<br>CS 145 B<br>CME 106 B+<br>DANCE 46 CR | units: 20 GPA: 3.78<br>CS 140 A<br>CS 155 B+<br>CS 247 A<br>CS 399 A+<br>GES 1 B+ | units: 8<br>MATH 53 5<br>CS 121 3 |
| Spring | units: 16 GPA: 3.56<br>CS 106B A-<br>CS 196 CR<br>IHUM 5B B+<br>PWR 1 A- | units: 18 GPA: 3.93<br>CS 103B A<br>CS 194 A+<br>PWR 2 A<br>EE 108B B+<br>COMM 169 A | units: 17 GPA: 3.91<br>CS 155 B+<br>CS 194 A+<br>CS 399 A<br>ENGR 50 A-<br>MUSIC 18B A+ | units: 0 |
| Summer | units: 0 GPA: N/A | units: 0 GPA: N/A | units: 0 | units: 0 |

Import
Transcript
Export
Adobe PDF

Grades
3.69 overall GPA
154 completed units
174 total units

GERs
Disciplinary Breadth:
Hum: MUSIC 18B
EngrAppSci: CS 106A
Math: CS 103A
NatSci: GES 1
SocSci: COMM 169

Education for Citizenshi
(2 of 4 required)
AmerCul: MUSIC 18B
GlobalCom: n/a
Gender: n/a
EthicReas: n/a

**Figure 1: CourseRank Screen Shots: course description (left), course planner (right).**

In this section, we drill-down in more detail into these services, to illustrate the types of challenges we address and the types of solutions that emerge from our work.

## 3.1 Flexible Recommendations

Social networking and commerce sites often provide recommendation services. For instance, MovieLens [12] recommends movies to watch, while Amazon [11] recommends books to buy. Recommendations are also important in CourseRank, and there are many more recommendation challenges than in a traditional site. In CourseRank, there are multiple dimensions to recommend (courses, quarters, majors, instructors), and there are multiple ways to recommend them. For example, a course recommendation can be based on what "similar" students have taken, where similarity is based on liking the same courses, or getting the same grades, or being in the same major. The recommendations can also be based on what courses are needed for graduation, or on what is available when the student has free time in the week. The same course can be offered at different times, with different instructors, teaching assistants, and textbooks. In addition, courses (unlike books or movies) often need to be taken in a certain order or must satisfy certain constraints.

For this purpose, we have developed a flexible recommendation service that allows recommendations to be easily defined, customized, and processed. The engine gives not one canned recommendation (as most current systems do [1]), but the flexibility to specify what is desired. Our goal is to allow a student to specify their goal (courses, quarters, instructors, ...), the basis of recommendations (grade similarity, evaluation similarity, ...) and filtering conditions (e.g., I am looking of a biology class that satisfies my science requirement).

A given recommendation approach can be expressed declaratively as a high-level workflow over structured data and then executed by the underlying engine. Our view is that a site administrator declaratively expresses a suite of workflows; then students can select a workflow, and provide parameters to it. For example, parameters may specify courses the student is interested in or peer students with whom to get compared.

We now describe our flexible recommendation model, FlexRecs (presented at the SIGMOD Conference [7].) Since CourseRank data is currently relational, we start with a relational algebra representation of workflows, containing *traditional relational operators* such as select, project and join, *plus new recommendation operators* that generate or combine recommendations. At the heart of these new operators is a special recommend operator that takes as input a set of tuples and ranks them by comparing them to another set of tuples. The operator may call upon functions in a library that implement common tasks for recommendations, such as computing the Jaccard or Pearson similarity of two sets of objects. The operator may be combined with other recommendation and traditional relational operators.

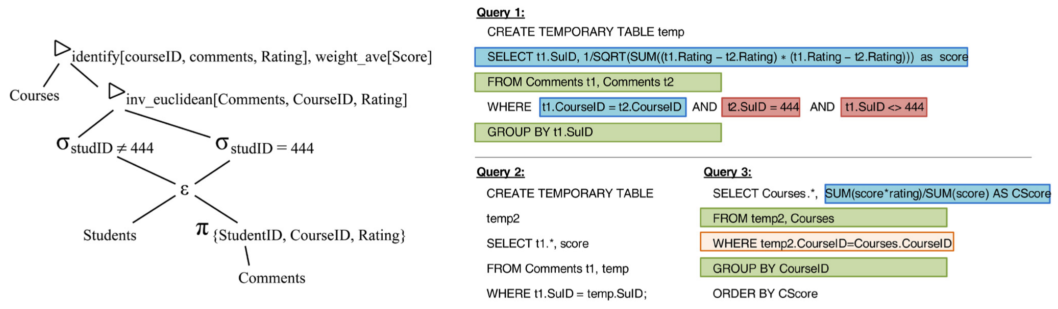
To illustrate FlexRecs, suppose that our information on courses, students and evaluations is stored in the following three relations. (Even though we continue to focus on academic planning, our model is generic and can be used in any recommendation scenario.)

Courses(<u>CourseID</u>, DepID, Title, Description, Units, Url)
Students(<u>SuID</u>, Name, Class, GPA)
Comments(<u>SuID</u>, <u>CourseID</u>, <u>Year</u>, <u>Term</u>, Text, Rating, Date)

In order to build recommendations, we have a library of comparison functions (e.g., to compare course ratings, course topics, student names, etc), such as Pearson's, and Jaccard index, and for aggregations, such as average and weighted average.

Assume that we want to compute course recommendations for a student with id 444 based on the ratings of similar students. In this case we assume that two students are similar if their course ratings are similar. For our example, we will compute similarity between two students by taking the inverse Euclidean distance of their course ratings. We compute the final course ratings by taking the weighted average of course ratings by students who are similar to student 444. The desired recommendations are captured by the workflow shown in Figure 2 (left). As we can observe, it is composed of traditional select, project, join operators, and it also contains some new operators that we describe below.

Ideally, we would like to represent our application entities with a single relation. For instance, a tuple in such a relation could contain base information on a student (e.g., name), plus the courses a student has taken. For this purpose, we use an *extend operator* ($\varepsilon$) that generates a virtual 2-level nested relation. This operator allows


▷ identify[courseID, comments, Rating], weight_ave[Score]
Courses
▷ inv_euclidean[Comments, CourseID, Rating]
σ studID ≠ 444
σ studID = 444
ε
Students
π {StudentID, CourseID, Rating}
Comments

```
Query 1:
  CREATE TEMPORARY TABLE temp
  SELECT t1.SuID, 1/SQRT(SUM((t1.Rating – t2.Rating) ∗ (t1.Rating – t2.Rating)))  as  score
  FROM Comments t1, Comments t2
  WHERE  t1.CourseID = t2.CourseID  AND  t2.SuID = 444  AND  t1.SuID <> 444
  GROUP BY t1.SuID
```

```
Query 2:                          Query 3:
  CREATE TEMPORARY TABLE            SELECT Courses.*, SUM(score*rating)/SUM(score) AS CScore
  temp2                             FROM temp2, Courses
  SELECT t1.*, score                WHERE temp2.CourseID=Courses.CourseID
  FROM Comments t1, temp            GROUP BY CourseID
  WHERE t1.SuID = temp.SuID;        ORDER BY CScore
```

**Figure 2: Sample workflow (left) and generated SQL plan (right).**

"extending" each tuple from one relation with the set of joining tuples from a different relation. In our example workflow, students are extended with their course ratings, so that the set of ratings for each student can be "viewed" as another attribute of the student by subsequent operators in the workflow irrespective of the database schema. Hence, it would be just as easy to compare students based on their name (a normal attribute) or based on their ratings (an extended attribute). In a sense, *extended relations* can be thought of as "views" that group together information related to an individual entity and represent it as a single tuple.

We observe that recommendations are based on comparisons (e.g., courses are rated against student ratings, students are compared to a student based on their ratings in order to find similar students, and so forth). For this purpose, we use the *recommend operator* ($\triangleright_{cf}$), which rates the tuples of a set by comparing them to the tuples of another set using a comparison function $cf$. Our example workflow has two recommend operators. The lower one finds similar students to the student with id 444 using the inverse Euclidean distance of their ratings. The upper one finds courses recommended by these students taking a weighted average of their ratings.

We have built a FlexRecs engine in Java that compiles and executes workflows on top of the CourseRank MySQL database. To illustrate, Figure 2 (right) shows the compiled plan (Queries Q1, Q2, Q3) for our sample workflow. We will first see the set of queries for implementing the lower recommend operator in Figure 2 (left). Query Q1 has several shaded parts: (a) the select operators have been included as conditions in the WHERE clause, (b) the recommend operator, which compares students using the inverse Euclidean comparison function on their course ratings, is implemented by combining the aggregation functions that are supported by the underlying database, (c) the extend operator is implemented by a GROUP BY clause. This query creates a temporary in-memory table that contains two attributes for each student: the student id and a score. Q2 combines for each student the score and the student ratings into one relation. Then, the higher recommend operator computes course recommendations based on the ratings provided by the similar users based on their scores. The computations required are again realized by leveraging the database's existing aggregation capabilities as shown in Q3.

Handling the full suite of FlexRecs operators is more challenging than what this simple example illustrates. In summary, FlexRecs is a way to express recommendation strategies more compactly and clearly than using, say SQL or Java. FlexRecs makes it possible to offer users (in any social site) a variety of recommendation strategies, that can be easily tailored to their interests. In CourseRank we have been able to quickly modify existing workflows to experiment with a variety of recommendation strategies. There is, of course, still much remaining work, for instance:

- *Optimization.* Implementing the new operators inside the database engine will enable the implementation of special workflow optimization schemes. For example, we may be able to push down selections and change the order of recommendation operators or dynamically define what comparisons should be performed in order to achieve a good trade-off between recommendation efficiency and effectiveness.
- *User Interface.* Developing appropriate user interfaces that will allow users to specify the kind of recommendations they want is also very challenging. We plan to develop and evaluate different interfaces for students to select workflows and provide parameters. Such an interface will allow users to specify their target (e.g., courses, instructors, majors, quarters), filtering conditions (e.g., biology courses, engineering majors), and the basis for the recommendation (students with similar grades, with similar tastes, and so forth).
- *Other Applications and Non-Relational Data.* We will explore how FlexRecs can be extended to support non-relational data (e.g., XML, Jason) and other recommendation applications.

## 3.2 Course Requirements

In order to graduate, students must satisfy a set of requirements. For example, a Computer Science (CS) major at Stanford must satisfy a set of sub-requirements, one of which is the math requirement (simplified):

- The student must complete Math 41 and 43, or as an alternative Math 19, 20, 21.
- The student must complete either CS 103X or the pair CS 103A, 103B.
- The student must complete two electives out of the set Math 51, 103, 108, ... CS 156, 157, .... Completion of Math 52 and Math 53 will together count as one Math elective. Restrictions: Math 51 and 103, or Math 51 and CME 100, or ... may not be used in combination to satisfy the Math electives requirement.
- The total units for Math courses should be 23 or greater.

A system like CourseRank needs to understand such relationships in order to (*a*) help students manage their courses (e.g., am I done with the foreign-language requirement?), and (*b*) improve recommendations (e.g., course $x$ is highly recommended for you because it helps you complete your major requirements faster).

Achieving this functionality we need: (*a*) a language for describing the requirements commonly seen at universities; (*b*) algorithms for efficiently checking if requirements have been satisfied (and for explaining what parts have not been satisfied); and (*c*) ways to translate our knowledge of what courses help a student complete requirements into recommendations for the student.

University requirements are quite diverse, so they represent the ideal testing ground to understand recommendations in the face of complex constraints. Simply capturing the requirements in a succinct and usable form is one of the challenges. (Several commercial products provide ways of capturing academic requirements, but their models are so complex that they are not widely used.) Furthermore, it turns out that efficiently checking requirement satisfaction is not trivial. One complexity is that a course $a$ may appear in multiple sub-requirements, and it can only be used to satisfy one of the sub-requirements (see example below). Furthermore, there are often exceptions to the rules, pre-requisites for courses, and so on.

In general, we have shown that checking such complex constraints is NP-hard [15]. However, we have identified a class of requirements that in practice is at the core of most actual requirements and that can be checked efficiently. We next illustrate this class and one efficient checking algorithm that can form the basis of a more general scheme.

In the class that we study, requirements are expressed as conjunctions of sub-requirements, where each sub-requirement $R_i$ is of the form `take` $k_i$ `from` $S_i$. Here, $S_i$ is a set of courses, and $k_i > 0$. Note that $S_i \cap S_j$ need not be empty, i.e., there could be courses that are common to the two sub-requirements as well, e.g., the database systems principles course could be both in the theory and systems sub-requirements. However, a taken course can be counted towards only one sub-requirement.

To illustrate requirements and our algorithm, say students must satisfy these three sub-requirements:

- $R_1$: `take 1 from` $\{a, p\}$
- $R_2$: `take 2 from` $\{p, d, i\}$
- $R_3$: `take 1 from` $\{i, o\}$.

A student who has taken courses $\{a, p, i, o\}$ has satisfied the requirement. Another student, say Bob, who has only taken $\{p, d, o\}$ has not satisfied the requirement.

Our checking (and recommendations) algorithm is based on building a flow graph, as illustrated in Figure 3(left). The courses are divided into two groups representing the courses taken by Bob (left top oval), and the remaining courses (left bottom oval). Each course is connected to the sub-requirements it helps satisfy. Each link has two numbers associated with it. The first is a maximum capacity. The capacity is 1 for all links except those connecting to the target $t$, in which case the capacity is the "take $k$" value associated with the sub-requirement. The second number is a cost (in square brackets), which is used only for links from the source $s$ to a course from the ones that Bob has not taken yet. For now, assume these costs are 1.

It turns out that if we run a min-cost max-flow algorithm on this graph we can not only check if Bob has satisfied the requirements, but we can also obtain the smallest set of additional courses that are needed towards this end. In particular, if there is a feasible flow of magnitude $\sum_j k_j$, then there is an assignment of courses to sub-requirements such that each sub-requirement is satisfied. (The converse is also true.) If no additional courses are used (i.e., have a non-zero flow in the solution), then Bob has satisfied the requirements. If not, then the used courses represent the smallest set of courses needed, i.e., courses we can recommend to Bob. In this example, Bob has only taken 3 courses, so some of the remaining courses are also needed. The algorithm uses course $i$, achieving a max-flow of 4 to $t$, at a min-cost of only 1. Thus, Bob is recommended course $i$. It can be shown [15] that the complexity of checking/recommending courses in this fashion is $O((c*r+m)^2 \sum_j k_j)$, where $c$ is the number of courses, $m$ is the number of sub-requirements, and each course appears in at most $r$ sub-requirements. Thus, this approach is relatively efficient.

We can take this approach one step further by assigning to all courses that have not been taken by a student costs that reflect their "inverse desirability". That is, using traditional recommendation schemes, we can assign to each course $c$ a score $sc(c)$ (between 0 and 1) that represents its utility based on grades of similar students, popularity, ratings, prerequisites being satisfied, etc. Then we can use $1 - sc(c)$ as the cost of a course, and the min-cost max-flow algorithm will give us a set of candidate courses that (*a*) contains the smallest feasible number of courses, and (*b*) among the smallest feasible sets, has the highest aggregate score. Thus, we can now recommend courses that both help meet requirements and are desirable. To complete our example, say given Bob's grades and tastes, course $a$ has a score of 0.9 (very desirable), while $i$ has a score of 0.5 (less desirable). In this case, the recommendation changes from $i$ to $a$. (When the costs were equal, $R_1$ was satisfied by $p$ and $R_2$ by $i$, $d$. Now, $R_1$ is satisfied by $a$ and $R_2$ by $p$, $d$.)

Our network flow solution can be extended to handle additional types of constraints [15]. It can also be used as an initial filtering step when more complex constraints exist. That is, we can generate solutions that satisfy the constraints we can handle efficiently, and then check if the resulting assignments of courses to sub-requirements also satisfy the more complex constraints. If the complex constraints are not met, then we can do more sophisticated (and expensive) searching (which we believe will be rare).

Clearly, there is still substantial work to be done:

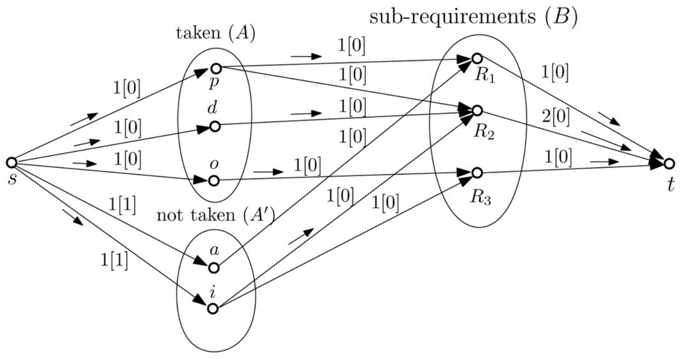
- *Recommendation Evaluation.* Using actual requirements from a variety of programs and the courses taken by students, we will determine how efficient each recommendation approach is.
- *Prerequisites.* We will incorporate prerequisites into our framework. For instance, we may not want to recommend course $a$ if course $b$ needs to be taken first, unless we can also incorporate $b$ into our recommendation.
- *Other domains.* We will apply/extend our requirement model and algorithms to other settings. For example, say a banker wants to recommend an investment portfolio to a customer, with constraints on the type of investments and amounts. What constraints appear here (perhaps similar to course constraints)? How can we make recommendations with such constraints?

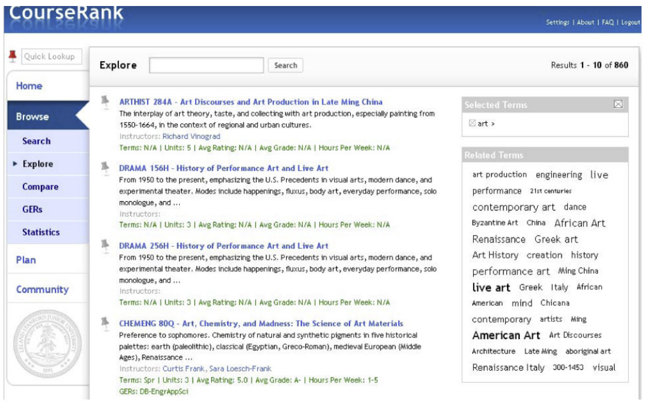
## 3.3 Course Cloud

In order to facilitate course planning, CourseRank offers two traditional interfaces: one for browsing courses based on department and a keyword-based search interface. Keywords are searched in the title and the description of courses.

When browsing courses based on department, students have to sift through long lists of courses and read their descriptions in order to discover courses of interest. Many courses may cover common topics and different departments may offer courses on similar topics making locating and sorting out the available options very tedious. On the other hand, keyword searching offers more flexibility but users still need to figure out the right search keywords, not always an easy task. Furthermore, users often want to search beyond the immediate course description. For example, if a student searches for "Java", he may be interested not only in courses that explicitly mention this word in their title or description, but also in courses with implicit references to "Java", such as in their comments.

*CourseCloud* (presented at EDBT 2009 [9]) is an improved search service, especially targeted at the *discovery of unexpected but useful* courses or other resources (as opposed to searching for a specific course with known characteristics). *CourseCloud* uses three main ideas: (*a*) in addition to search results, the service presents a "tag cloud" where users can see unexpected terms that may be of interest. In this case, the "tags" are not traditional tags added by users,





Results **1** - **10** of **860**

ARTHIST 284A - Art Discourses and Art Production in Late Ming China
The interplay of art theory, taste, and collecting with art production, especially painting from 1550-1664, in the context of regional and urban cultures.
Instructors: Richard Vinograd
Terms: N/A | Units: 5 | Avg Rating: N/A | Avg Grade: N/A | Hours Per Week: N/A

DRAMA 156H - History of Performance Art and Live Art
From 1950 to the present, emphasizing the U.S. Precedents in visual arts, modern dance, and experimental theater. Modes include happenings, fluxus, body art, everyday performance, solo monologue, and ...
Instructors:
Terms: N/A | Units: 3 | Avg Rating: N/A | Avg Grade: N/A | Hours Per Week: N/A

DRAMA 256H - History of Performance Art and Live Art
From 1950 to the present, emphasizing the U.S. Precedents in visual arts, modern dance, and experimental theater. Modes include happenings, fluxus, body art, everyday performance, solo monologue, and ...
Instructors:
Terms: N/A | Units: 3 | Avg Rating: N/A | Avg Grade: N/A | Hours Per Week: N/A

CHEMENG 80Q - Art, Chemistry, and Madness: The Science of Art Materials
Preference to sophomores. Chemistry of natural and synthetic pigments in five historical palettes: earth (paleolithic), classical (Egyptian, Greco-Roman), medieval European (Middle Ages), Renaissance ...
Instructors: Curtis Frank, Sara Loesch-Frank
Terms: Spr | Units: 3 | Avg Rating: 5.0 | Avg Grade: A- | Hours Per Week: 1-5
GERs: DB-EngrAppSci

**Figure 3: Example flow graph (left); Example Course Cloud (right).**

but terms from the database that are explicitly or implicitly connected to the search results. (*b*) CourseCloud searches for terms (keywords entered by user or tags) in data related to courses (e.g., student comments), not just in the course records. (*c*) CourseCloud uses the tags for navigation and search refinement. Our initial experience with this prototype shows that for some students it provides a very useful service.

Figure 3(right) illustrates the CourseCloud interface, after the student has typed in the keyword "art". The left display shows courses related to the search, that have "art" in their description or in "nearby" records (e.g., in comments). On the left is the tag cloud, providing many diverse concepts related to "art" that are found in the matching courses, such as "performance", "art production", and "Renaissance". For example, the term "performance" is found in many user comments that refer to "art" courses with live performances. The data cloud conveniently categorizes courses in a digestible way under different concepts. Thus the student can find out that there are courses offered not only by the ART HISTORY program (identified by the course code in the results) but also from other programs that address other aspects, such as the DRAMA or HUMANITIES programs.

When the student clicks on a tag, say "architecture", the system adds the term to the search and displays new results and a new cloud, allowing the student to drill-down. Some of the new tags, like "Byzantine art" or "religious art" may be unexpected to the student, allowing the discovery of courses the student might not have thought of.

Even though tag clouds are popular on some web sites, there has been little research on them, and to our knowledge, no work on using them to explore non-tagged content. There are many important questions to investigate related to this type of service:

- *Tag selection.* In [9] we explore some initial techniques that show promise in identifying the terms users find most useful. There are many other options that need to be evaluated in terms of coverage, diversity, overlap, and other aspects.
- *Personalization.* Displaying the most popular tags is not hard, but displaying tags that are "personalized" to a user is much more challenging. We will study the performance of various tag selection schemes that adapt to the needs and preferences of a particular user.

## 4. SOCIAL IMPACT

In a social networking site, users interact not just with the computer, but with each other. CourseRank provides a living laboratory for studying or revisiting human interactions in the specific context of an electronic academic community.

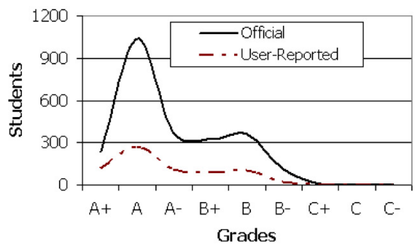
We briefly summarize two results we have obtained using CourseRank. Details on how these two results were obtained, plus additional results can be found in [8], published in ICWSM 2009.

*Are users of social sites truthful?* Users at social sites provide a lot of information about themselves, e.g., their gender, age, interests, and so on. How reliable is all this information? There is seldom a way to verify the authenticity of user provided data, and anecdotally we know the some users lie about their age or gender. However, CourseRank gives an opportunity to verify some of the information users provide. For example, Figure 4(left) shows two grade distributions for students in the Engineering School at Stanford. The top one is the official distribution, provided to us by the registrar. It shows how many students received a particular grade in a period. The bottom distribution uses self-reported grades, which the students enter into CourseRank as they plan their academic program. We have fewer self-reported grades, but overall the distributions follow very similar patterns. This result suggests that on aggregate users are giving us very accurate grade information. Of course, users have an incentive to give good data, since the data helps them plan their program. Additional experiments are needed to understand if there is a grade bias in some circumstances (e.g., in popular or large courses) or for other type of information.

*Are raters objective?* Many universities use course evaluations to evaluate and promote professors, and university administrators frequently argue that such evaluations are unbiased. That is, a professor cannot improve his ratings by giving out higher grades. However, Figure 4(right) shows a clear correlation between the grade a student receives and the rating he gives to the course. For each possible grade (horizontal axis), the vertical axis shows the average course rating given by students that earned that grade. (User ratings range from the lowest rating of 1 to the highest of 5.) In this case the grades are self reported, as discussed above. Of course, this result does not prove that instructors can improve their ratings by giving out higher grades, it only shows a correlation. But we plan to shed more light on this issue by examining the written comments left with the course evaluations, and by comparing the ratings and grades of a student in different courses.

There are several interesting questions to study, such as:

- *Community effect*. What role does community size and corpus size play in user behavior? In CourseRank, we have communities with varying characteristics (e.g., departments with few or many students, departments that offer many or few courses). Do students in different communities behave differently?



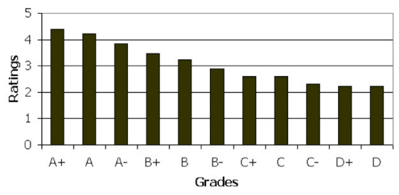


**Figure 4: Sample results: Truthfulness (left); Objectiveness (right).**

- *Incentives*. How can we incentivize students to evaluate more courses, or to give comments that are more constructive? In the early stages of CourseRank, we raffled an iPod to get users to try the site. Can similar techniques be used? What is their impact?

## 5. CONCLUSIONS AND FUTURE WORK

We will continue to use CourseRank as a live testbed for fundamental research into social systems, developing and evaluating the algorithms and services that drive such systems. There are many interesting questions that can be addressed, regarding how students use the site, the veracity of user provided data, the usefulness of our course recommendations, and so on. At the same time, CourseRank has been spun out of our lab as a company, and the system is being deployed at other universities.

One of the strengths of our project is that we have a concrete application (student academic planning) with many actual users as well as rich and interesting data. In spite of our focus on this application, we believe that much of our work will be applicable to other applications, and indeed, a concurrent goal of our project is to explore other domains and applications. For instance, several companies have expressed interest in using CourseRank as a starting point for a "corporate social site" where employees discover resources and plan projects. Many of the services we explore (e.g., recommendations) would be applicable in such a system.

## 6. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[2] C. Brooks and N. Montanez. Improved annotation of the blogosphere via autotagging and hierarchical clustering. In *Proceedings of the 15th International Conference on World Wide Web*, 2006.

[3] Del.icio.us: url: http://del.icio.us/.

[4] Flickr: url: http://www.flickr.com/.

[5] S. Golder and B. A. Huberman. Usage patterns of collaborative tagging systems. *Journal of Information Science*, 32(2):198–208, 2006.

[6] V. Gomez, A. Kaltenbrunner, and V. Lopez. Statistical analysis of the social network and discussion threads in slashdot. In *Proceedings of the 17th International Conference on World Wide Web*, 2008.

[7] Georgia Koutrika, Benjamin Bercovitz, and Hector Garcia-Molina. Flexrecs: Expressing and combining flexible recommendations. In *SIGMOD Conference*, 2009.

[8] Georgia Koutrika, Benjamin Bercovitz, Filip Kaliszan, Henry Liou, and Hector Garcia-Molina. Courserank: A closed-community social system through the magnifying glass. In *Third International Conference on Weblogs and Social Media (ICWSM)*, 2009.

[9] Georgia Koutrika, Z. Mohammadi Zadeh, and Hector Garcia-Molina. Data clouds: Summarizing keyword search results over structured data. In *12th International Conference on Extending Database Technology (EDBT)*, 2009.

[10] X. Li, L. Guo, and Y. Zhao. Tag-based social interest discovery. In *Proceedings of the 17th International Conference on World Wide Web*, 2008.

[11] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, Jan/Feb 2003.

[12] B.N. Miller, I. Albert, S.K. Lam, J.A. Konstan, and J. Riedl. Movielens unplugged: Experiences with an occasionally connected recommender system. In *IntŠl Conf. Intelligent User Interfaces*, 2003.

[13] G. Mishne. Autotag: collaborative approach to automated tag assignment for weblog posts. In *Proceedings of the 15th International Conference on World Wide Web*, 2006.

[14] T. Ohkura, Y. Kiyota, and H. Nakagawa. Browsing system for weblog articles based on automated folksonomy. In *Proceedings of the WWW 2006 Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics*, 2006.

[15] Aditya Parameswaran, Petros Venetis, and Hector Garcia-Molina. Recommendation systems with complex constraints: A courserank perspective. In *Stanford InfoLab Technical Report, available at http://ilpubs.stanford.edu:8090/909/*, 2009.

[16] P. Schmitz. Inducing ontology from flickr tags. In *Collab. Web Tagging Workshop in conj. with WWW2006.*

[17] technorati: url: http://www.technorati.com/.

[18] Z. Xu, Y. Fu, J. Mao, and D. Su. Towards the semantic web: Collaborative tag suggestions. In *Collab. Web Tagging Workshop in conj. with WWW2006.*

[19] J. Zhang, M.S. Ackerman, and L. Adamic. Expertise networks in online communities: Structure and algorithms. In *WWW*, 2007.

# On Energy Management, Load Balancing and Replication

Willis Lang     Jignesh M. Patel     Jeffrey F. Naughton
Computer Sciences Department
University of Wisconsin-Madison, USA
{wlang, jignesh, naughton}@cs.wisc.edu

## Abstract

In this paper we investigate some opportunities and challenges that arise in energy-aware computing in a cluster of servers running data-intensive workloads. We leverage the insight that servers in a cluster are often underutilized, which makes it attractive to consider powering down some servers and redistributing their load to others. Of course, powering down servers naively will render data stored only on powered down servers inaccessible. While data replication can be exploited to power down servers without losing access to data, unfortunately, care must be taken in the design of the replication and server power down schemes to avoid creating load imbalances on the remaining "live" servers. Accordingly, in this paper we study the interaction between energy management, load balancing, and replication strategies for data-intensive cluster computing. In particular, we show that Chained Declustering – a replication strategy proposed more than 20 years ago – can support very flexible energy management schemes.

## 1 Introduction

Servers consume tremendous amounts of energy, and the energy cost as a component of the TCO is quickly rising [4, 7]. In addition, servers often run at low utilization, typically in the 20-30% range [3]. This low utilization suggests that one way of saving energy is to selectively power down servers. However, arbitrarily powering down servers that are running data-intensive applications is problematic, as it can render a portion of the data unavailable.

Fortunately, most clusters servicing data-intensive workloads already employ data replication schemes, to ensure data availability and reliability in the presence of failures. One of our key observations is that *this same replication can be exploited to ensure availability in the presence of deliberate server power downs intended to save energy.* However, while data replication can indeed be exploited to power down servers without losing access to data, care must be taken in the design of the replication and server power down schemes to avoid creating load imbalances on the remaining "live" servers, which can have severe performance consequences.

To see this point, consider a system that uses the common mirroring replication strategy. To make this example more concrete, suppose that there are four nodes using mirrored replication, where each data partition is stored in exactly two different storage units. In addition, suppose that the data set is split into two partitions, $P_0$ with mirror $R_0$, and $P_1$ with mirror $R_1$. Assume that node $n_0$, $n_1$, $n_2$, and $n_3$ store $P_0$, $P_1$, $R_0$, and $R_1$ respectively. Furthermore, assume that queries can be sent to either the primary copy or the replica for load balancing. If the overall system utilization is at or below 50% of the provisioned utilization, then nodes $n_2$ and $n_3$ could be turned off to save energy, while nodes $n_0$ and $n_1$ would then operate at 100% of the provisioned utilization. This is an ideal scenario and may be sufficient for certain systems. However, we wish to explore powering down nodes when utilization is between $50 - 100\%$ for a finer grained energy management scheme.

Now, consider another scenario in which the four nodes each initially see a load of 75%. The system has the capacity to run this workload on only three processors. Furthermore, by exploiting replication, we can certainly turn off one processor and still maintain access to all data.

Unfortunately, if we turn off node $n_3$, then nodes $n_0$ and $n_2$ will continue to operate at 75% of the provisioned utilization, but now both node $n_1$ and node $n_3$'s original load will be directed at node $n_1$, so the presented load there will be 150%, and the system will likely fail to meet its performance requirement. Such large load imbalances may be acceptable in certain environments, but the performance degradations are usually unacceptable (see Sections 2.1 and 3.1 for more details).

Given this example, our goal is to investigate the interaction between replication and power down schemes to provide the foundation for energy management approaches that gracefully adapt to overall system utilization. This should be done in such a way as to maximize energy efficiency by powering down some nodes while ensuring that the utilization of the remaining nodes does not exceed a targeted peak utilization.

Given the many decades of work on designing replication schemes, the immediate question is whether or not there is a replication scheme that fits our goals of producing balanced, and energy-efficient cluster management strategies. As we will demonstrate, the surprising answer is yes — one of the earliest proposed parallel database data replication schemes, the "Chained Declustering" technique [10], when coupled with careful choices of which nodes to power down, can be exploited to achieve the above goal. In this paper, we present and evaluate two Chained Declustering-based schemes that differ in they way they power down/up nodes in a cluster.

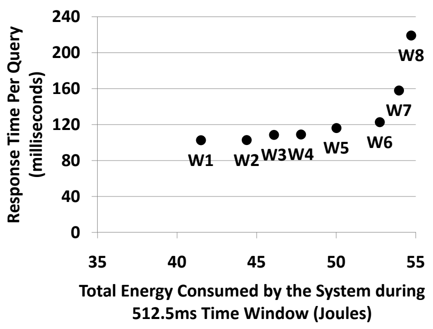


**Figure 1. Energy consumption and response time profile**

To the best of our knowledge this is the first paper exploring this interaction between power down sequences and replication strategies while controlling load imbalances.

## 2 Background and Problem Specification

In this paper, we use the term *load* on a node to refer to the work that is being carried out on a node. In a system with a number of concurrent queries, each with the same processing cost, the load can simply mean the number of queries per node. The term *utilization* of a server node refers to the resource consumption on the node. The term *overall system utilization* refers to the average utilization across all the server nodes in the system. *Maximum node utilization* refers to the maximum utilization across all the server nodes.

Often cluster systems are designed to handle a certain provisioned *peak* load. We refer to the utilization using a value expressed as a percentage. Within this context, a utilization of 100% simply refers to operating at an initial designated "peak load" (which could be lower than the system's peak load at which it is stable). Lower utilization values, e.g., 50%, imply a corresponding reduction in the load (and an increase in server idle time).

The energy management schemes that we describe in this paper work by taking some nodes *offline*, which refers to a node being powered down to save energy. Nodes that are available to run queries are *online*. An offline node becomes available when it is powered up, in which case it then comes online. (In the more traditional case of replication for failure management, offline refers to the node being unavailable due to some component failure.)

Finally, an operational state for the entire system is defined as: The **operating state** of the entire system, $s(m)$, is a state where $m$ of the $N$ total nodes in the system are offline.

### 2.1 Server Load vs. Energy Consumed

As pointed out in [3], the relationship between the load on a server and the energy consumed by the server is not linear. As an example, consider Figure 1, which shows the characteristics of a 1% clustered index query workload running on a commercial DBMS. (See Section 5.2 for more details about this workload.) In this graph, the point W1 corresponds to a server workload in which one instance of the query takes X ms to run followed by the server being idle for 4X ms. One can view this workload as a series of time windows, each of size $5X$ ms, where $X$ is the time to run the query. For workload W1, only one query is run in each window.

Other points in this graph correspond to higher server utilizations, which we achieved by randomly adding more queries in the time window (of length 5X ms), thereby reducing the idle component. Specifically, a point W$i$ corresponds to injecting $i$ queries, with random arrival times, into each $5X$ ms time window. Figure 1 shows for each workload the average execution time per query and the energy consumed by the server to run the workload.

Now, consider the point W1 in Figure 1. In this case, the server consumes about 41.5 Joules and provides a query response time of 102.5 ms. Most of this energy, specifically 74%, is consumed while the server is idle. As we add more queries to the workload, i.e., go beyond W1, the idle time decreases and a larger fraction of the energy consumed by the server is spent actually running the queries. At W5, since each query takes X ms to run, we are running at some provisioned "peak" utilization of 100%. Notice how performance rapidly degrades beyond W6. Operating at such points (W7 and beyond) merely to save power may be unacceptable as this region likely represents an unstable operating range.

If efficiency is defined as the energy consumed by the server per query, of the five workloads W1 to W5, W5 has the highest efficiency. Notice, however, the response time per query is slightly worse at W5 than at the other four points, since at the other points there is less contention for resources across different queries.

Thus we have two possibly conflicting optimization goals. The first is the traditional one — we could simply optimize for response time, which means running the system at point W1. However, typically in data center environments, the performance constraint to meet is not "as fast as possible;" but rather, something more like "no worse than $t$ seconds per query for this workload." When agreeing to such Service Level Agreements (SLAs), data center service providers tend to be conservative and agree to performance that they can generally guarantee under the heaviest provisioned load, rather than performance they can meet in the best case. Consequently, the second optimization goal, and the one that we focus on in this paper, is to reduce the energy consumption while staying below a response time target.

### 2.2 Problem Statement

We want an energy management scheme that starts with an operating state $s(m)$ for a system with maximum node utilization of $u$ ($u < U$). Here $U$ refers to some maximum tolerable system utilization (perhaps defined by an SLA). We want the system to move to a new operating state $s(m')$ with maximum node utilization $u'$ such that $u' < U$ and $m \leq m'$, and at least one copy of each data item is available on the remaining servers that are still powered up.

Note that $U$ is defined relative to the initial designated peak load (cf. Section 2). Consequently, $U$ can be greater than 100%; e.g., if the maximum tolerable response time is 120*ms* in Figure 1, then $U$ is 120% (at W6).

Notice that the problem statement also allows setting U to 100%, in which case no node operates over the designated peak capacity.

In addition, we require "data availability" – i.e., the power down sequence does not deliberately make any data item unavailable on the live servers that are powered up.

| Nodes: | $n_0$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ | $n_6$ | $n_7$ |
|---|---|---|---|---|---|---|---|---|
| Primary: | $R_0$ | $R_1$ | $R_2$ | $R_3$ | $R_4$ | $R_5$ | $R_6$ | $R_7$ |
| Backup: | $r_1$ | $r_2$ | $r_3$ | $r_4$ | $r_5$ | $r_6$ | $r_7$ | $r_0$ |
| Load: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Table 1. An 8 node CD ring without failure.**

| Nodes: | $n_0$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ | $n_6$ | $n_7$ |
|---|---|---|---|---|---|---|---|---|
| Primary: | — | $R_1(1)$ | $R_2(\frac{6}{7})$ | $R_3(\frac{5}{7})$ | $R_4(\frac{4}{7})$ | $R_5(\frac{3}{7})$ | $R_6(\frac{2}{7})$ | $R_7(\frac{1}{7})$ |
| Backup: | — | $r_2(\frac{1}{7})$ | $r_3(\frac{2}{7})$ | $r_4(\frac{3}{7})$ | $r_5(\frac{4}{7})$ | $r_6(\frac{5}{7})$ | $r_7(\frac{6}{7})$ | $r_0(1)$ |
| Load: | 0 | $\frac{8}{7}$ | $\frac{8}{7}$ | $\frac{8}{7}$ | $\frac{8}{7}$ | $\frac{8}{7}$ | $\frac{8}{7}$ | $\frac{8}{7}$ |

**Table 2. An 8 node CD ring with 1 failure.**

The schemes that we present differ in the "variance" in the load across the different nodes. In other words, some schemes result in larger variation in the loads across the nodes (cf. Section 5.4, Figure 6). While load variance (imbalances) are inevitable, and minor load imbalances do not create a problem, artificially creating major load imbalances can result in the system failing to meet its targeted performance (e.g., W7 and W8 in Figure 1). Accordingly, we require that the energy management techniques bound the load imbalances ($U$) that they introduce. Some thoughts on picking appropriate values of $U$ are presented in [11].

Finally, for certain system states, the nodes can be "perfectly balanced" – which means that each online node has the same node load. We discuss this further in Section 4.2.3.

## 3 Replication Revisited

Replication schemes are traditionally designed to allow continued access to data when some nodes fail. Here, we want to exploit replication for a related but different purpose: namely, allowing continued data access not when nodes fail, but when they are deliberately powered down to save energy, while controlling the resulting load imbalance. When we look at the commonly used techniques, such as RAID [14], Mirrored Disk [5, 6], and Interleaved Declustering [20], we find that they all produce undesired load imbalances as nodes become inoperable or do not allow us to turn off multiple nodes. For instance, Interleaved Declustering retains load balance when one node fails but loses data availability if any additional nodes are lost. Our goal here is to leverage a replication scheme to safely and easily power down any number of nodes for energy efficiency, and exploit the load balancing and failover properties of replication.

Dealing with updates in this environment poses certain challenges, but our schemes can be adopted to handle updates, as discussed in [11].

### 3.1 Mirroring Replication

The basic principle used in mirroring [5, 6] is to make a second copy of the data and store it on a different storage device. Then, when some disk fails, the load on the remaining copies goes up dramatically. For example, consider a 2X replication scheme, in which we have a primary copy and one additional replica. Then, when a disk with either of these copies fails, all the load from the failed disk is transferred to the remaining disk, thereby doubling the load on the remaining disk. If a 2X increase in load is unacceptable, then with mirroring there is no energy savings if the system load is between 50 and 100%.

Also notice that with mirroring, there are only two operating states, 100% online nodes or 50% online nodes, which implies that it can't effectively adapt to loads in between these two extremes.

### 3.2 Chained Declustering (CD)

Chained Declustering [10] is a replication scheme that stripes the partitions of a data set two times across the nodes of the system, thereby doubling the amount of required disk space. The main hallmark of this scheme is its tolerance to multiple faults along the chain, if those faults do not occur on adjacent nodes. Furthermore, along with high availability, the arrangement of the replicas along the chain allows for balanced workload distribution when some nodes are offline. If one thinks of all the nodes in the system as being arranged in a ring or chain, then Chained Declustering (CD) places a partition and its replica in adjacent nodes in the chain.

As an example of CD, consider a data set $R$, spread over 8 nodes in Table 1. Here the primary copies of the data set are $R_0$ ... $R_7$. The corresponding replicas are shown as $r_0$ ... $r_7$. The nodes $n_0$ ... $n_7$ are conceptually organized in a ring. Primary copy $R_i$ is placed on node $i$ and its replica $r_i$ is placed on the "previous" node. During normal operation, if the access to all the partitions is uniform, then the queries simply access the primary partitions.

Now consider what happens when a node is taken offline by our energy management methods. Table 2 shows what happens when node $n_0$ is offline. Since node $n_0$ holds the partition $R_0$, all queries against this partition must now be serviced by node $n_7$, which holds the only other copy of this partition. But simply redirecting the queries against partition 0 to node $n_7$ could double the load on node $n_7$. CD solves this problem by redistributing the queries against partition 7 across both copies of that partition's data, namely $R_7$ and $r_7$. It does this for all the partitions, and ends up with a system in which each node is serving the same number ($\frac{8}{7}^{th}$ of the original load) of queries, and hence is a *balanced* system.

While Table 2 shows what happens when one node is offline, CD can allow up to $N/2$ alternating nodes to go offline, where $N$ is the number of nodes in the system. We exploit this property of CD to develop various energy management schemes.

## 4 Using Replication for Energy Management

While CD can tolerate a variety of configurations with nodes being offline, as we show below, some of these configurations lead to system load imbalances. The protocol that is used to take nodes offline directly determines the uniformity and balance of the load on the remaining online nodes.

For the discussion below, we introduce a few additional terms: a *ring* refers to the logical ordered arrangement of all the nodes in a CD scheme. When a node in a ring goes offline, the ring is *broken* and produces a *segment*. Additional node failures partition segments into other segments. Each segment has two *end nodes*.

A key observation is that if the **ring** or a **segment** of a CD set of nodes is broken, then the two new end nodes of the resulting segment(s) are **essential**, where the term essential for a node implies that removing that node makes the data unavailable. Thus, to take nodes offline any scheme must select additional nodes from the remaining online nodes that are not end points of the remaining segments. Next, we present two protocols for selecting which nodes to take offline.

## 4.1 Dissolving Chain (DC)

Using the key observation described above, the DC protocol sequentially withdraws nodes so that data is always available. DC starts with a full ring of nodes online, and when it takes the first node offline, it produced two segments of equal (or nearly equal) length. The next node it takes down is the middle node in the longest remaining segment.

At any given point in time, DC has a number of segments that it keeps sorted based on the segment length. Its powering down algorithm is then simple – simply take the middle node down in the current longest segment. More details about this method, including pseudocode and the node powering up sequence can be found in [11].

## 4.2 Blinking Chain (BC)

The general intuition behind the Blinking Chain (BC) methods is to allow more general cuts than the simple binary cuts used by DC to: a) to reduce the variation in the load across the nodes that are still up, and b) produce states where the load across the nodes is "balanced".

For example, for a system where $N = 40$ nodes, for a DC system at $s(9)$, there will be segments of length $4, 2, 1$ with 28 nodes at $(5/4)$ load, 2 nodes at $(3/2)$ load, and 1 node at double load. A better way to cut the $N = 40$ ring results in 4 segments of length 4 and 5 segments of length 3. This results in minimal load variation across the remaining online nodes (the benefits of this are shown in Section 5.4). We now discuss how to create these segments.

### *4.2.1 Segments and Transitions*

Notice that in DC once a node is powered down, that node continues to remain powered down if utilization decreases monotonically. This strategy can result in long segments, which in turn implies bigger variation in loads across the online nodes. The main intuition behind BC is to reduce these load variations by allowing powered down nodes to be brought online to make the current segments more uniform in terms of their lengths (which leads to lower load variations).

Consider transitioning from a state with $m$ nodes offline to $m'$ nodes offline. A method to implement this transition is to bring all but the root node back online and then turn $m' - 1$ of them off, but this results in a high transitioning cost as each transition requires making $m + m' - 2$ node state changes (i.e., changing the state of a node from offline to online, or vice versa). These state changes can consume a significant amount of energy, and we would also *like to minimize the energy spent in making these transitions*. An interesting property of BC is that when transitioning from state $s(m)$ to $s(m')$, there may be offline nodes in the $s(m)$ configuration that can remain offline in the $s(m')$ configuration. By not changing the status of these nodes, the transitions can be made more energy efficient, as discussed next.

### *4.2.2 Optimizing the Transitions*

First consider finding states that provide the most "efficient" transitions, which implies making the least number of node state changes in the transition. In BC, the most efficient transition between two states $s(m)$ and $s(m')$ is such that only $|m - m'|$ nodes undergo transition. This efficient transition is defined formally as:

DEFINITION 4.1. *The **Optimal Blinking Chain Transition** $s(m)$ to $s(m')$ only requires $|m - m'|$ nodes to undergo transition.*

Details about how to implement this optimal transition can be found in [11]. The following proposition highlights a key relationship between divisible states ($s(m)$, $s(m')$ such that $m|m'$ or $m'|m$) and the Optimal Blinking Chain Transition.

PROPOSITION 4.1. *$s(m)$ to $s(m')$ is an optimal Blinking Chain transition iff $(m|m'$ OR $m'|m)$*

See [11] for the proof.

Proposition 4.1 tells us that in a given operating state, $s(m)$, for $N$ CD nodes, we can transition to another $s(m')$ with maximum efficiency if and only if $m'$ is a multiple or factor of $m$. While the Optimal BC Transition has interesting properties, it does not handle all possible state transitions. Specifically, it does not cover transitions between any states $s(m)$ and $s(m')$ when $m$ and $m'$ do not divide each other. For example, if $N = 42$, we cannot execute $s(6)$ to $s(15)$, since the optimal transition is not defined in this case.

To handle transitions between any two arbitrary states, we need a **General Blinking Chain Transition**. This transition is implemented as a composition of two Optimal BC Transitions: $s(m)$ to $s(GCD(m, m'))$ to $s(m')$, which maximizes the number of offline nodes that are untouched during the transition.

Using our previous example, if $N = 42$ and we wish to transition from $s(6)$ to $s(15)$, then using the General BC Transition, we can save 4 node transitions by doing two optimal transitions: one from $s(6)$ to $s(3)$ and the second from $s(3)$ to $s(15)$. Details about implementing the Optimal BC Transition can be found in [11].

Notice that BC transitions are "optimal", when only $|m - m'|$ nodes transition. Recall this is *always* the case for DC transitions. The implication of this property is discussed in Section 5.5.

### *4.2.3 Number of Balanced States*

Now consider the special states $s(m)$ where $m|N$, in which all the nodes have identical loads. We call such states **balanced** states, and denote this as $\bar{s}(m)$.

We can calculate the total number of possible balanced operating states for a Chained Declustered system of $N$ nodes by recognizing that the number of primes in the factorization of $N$ is what determines the number of balanced states. That is, we first factor $N$ as $N = p_1^{N_1} p_2^{N_2} ... p_j^{N_j}$, where $p_i$ is the $i^{th}$ prime number. Then, by simple combinatorics, the total number of unique factors of $N$ is $\Pi_{1 \le i \le j}(N_i + 1)$, which is also the number of balanced states for this system since $\bar{s}(0)$ replaces factor $N$.

## 5 Evaluation

To evaluate our proposed methods, we took an actual server and ran two prototypical workloads on the server. We then took actual measurements for both energy and response time on this server, as we varied the load on the server (i.e.,

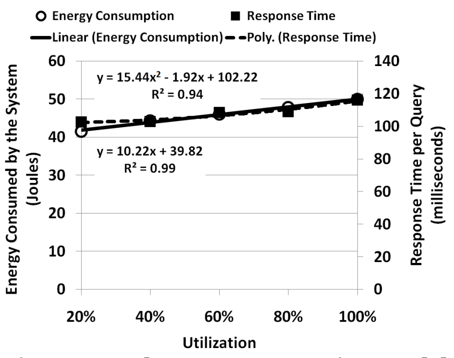


**Figure 2. Index query regression model**

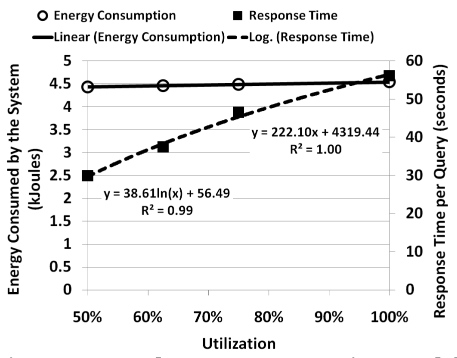


**Figure 3. Database scan regression model**

changed the server utilization). We then produced a model for a single node in a system. This model was then plugged into a larger model for the entire distributed system.

In all results presented below, we consider a system with 1000 nodes ($N = 1000$).

## 5.1 Experimental Setup

Our system under test (SUT) consisted of an ASUS P5Q3 Deluxe WIFI-AP motherboard with an Intel Core2Duo E8500, 2GB Kingston DDR3 memory, an ASUS GeForce 8400GS 256M graphics card, and a WD Caviar SE16 320G SATA disk. The power supply unit was a Corsair VX450W. System energy draw was measured using a Yokogawa WT210 unit as suggested by [1].

We ran queries on a commercial DBMS against a Wisconsin Benchmark (WB) table [9]. Client applications accessing the database were written in Java 1.6 using the JDBC connection drivers for the commercial DBMS.

We ran each experiment five times, and report the average of the middle three results. The ACPI S4 state was used as the offline state. Other offline states are discussed in [11].

## 5.2 Workload

We model two different types of workloads. The first workload uses WB Query 3. This query is a 1% selection query using a clustered index on a table with 20$M$ tuples (approx. 4GB table size). The actual workload consists of 1000 such queries with randomly selected ranges. This workload is used to model simple lookup queries. Our second workload is a file scan on a WB table (of varying sizes) that has no indices. This workload mimics queries that require scanning tables in a DSS environment. These workloads are described in more detail below.

### *5.2.1 Index Queries Workload*

To simulate varying node underutilization with the indexed range query, we defined various workloads for the indexed query by varying idle times (this is the same setup as described in Section 2.1). First, we ran this query and measured the query runtime. Lets call this X seconds. Then, we defined a 20% utilization workload as one in which the query runs for X seconds followed by an idle time of 4X seconds. In this setup, the server is presented with a series of these 5X time windows. An actual run consists of 1000 such windows, with random arrival time for the query in each window. We average the results over each run. Workloads with higher utilization are generated by injecting additional queries in this 5X window. For example a workload with 40% utilization has two queries in each 5X window, and a workload with 100% utilization has 5 queries in each 5X window.

To determine the value of X above, we ran 10000 random 1% selection queries and measured the average response time at 102.5 ms, with a standard deviation of 0.46 ms.

### *5.2.2 Database Scan Workload*

We modeled utilization of the system running scan workloads slightly differently to mimic a scenario in which a single scan runs across all the nodes in the system. In this case, when nodes are taken offline, the remaining online nodes have to scan larger portions of the data. In this model, let the time it takes a node to scan a 20$M$ tuple WB table be 56.49 seconds. This node is operating at 100% utilization, scanning as much as possible. For 75% utilization, we ask the node to scan a 15$M$ tuple table every 56.49$s$. Thus, over time, it is doing 75% of the work that it would do in the 100% case. Similarly, for 50% utilization, we ask it to scan a 10$M$ tuple WB table every 56.49$s$. Energy consumption is measured for the entire 56.49$s$ window. With increased utilization, the increase in response time increases (nearly) linearly. All scans are “cold” and there is no caching between successive scans.

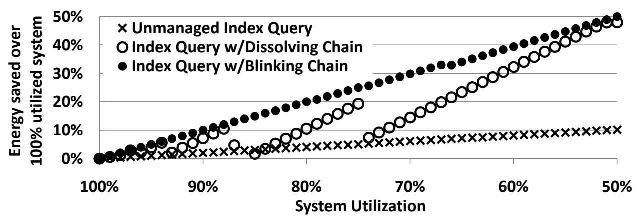
## 5.3 Modeling Energy and Response Time

In this section we present the measured energy consumption and response time results for each workload. We then use these results to develop a model for the behavior of a node in the system. All models were picked by trying a number of different linear and polynomial regression models, and picking the one with the lowest coefficient of determination, $R^2$. All presented models had $R^2 > 0.94$.

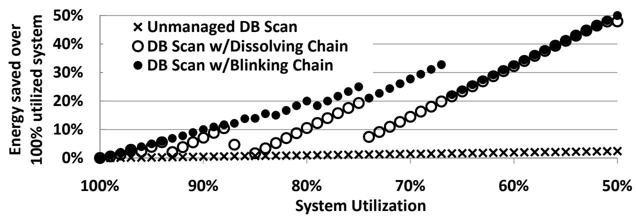
### *5.3.1 Indexed Query Workload*

The response time and energy measurement results for the index workload are presented in Figure 1 (in Section 2.1). Figure 2 plots this data with utilization on the x-axis, system energy consumption (for a 5X window) on the primary y-axis, and the query response time in milliseconds on the secondary y-axis.

Figure 2 also show the derived regression models for the average energy consumed by our SUT and average query response time as a function of utilization. The energy consumption model is linear while the response time model is quadratic.

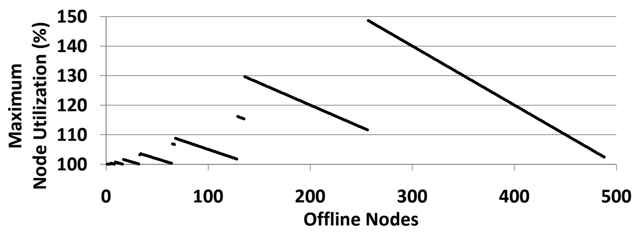


(a) Index Queries, N=1000

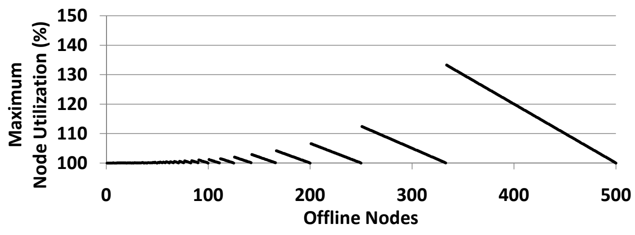


(b) Scan Queries, N=1000

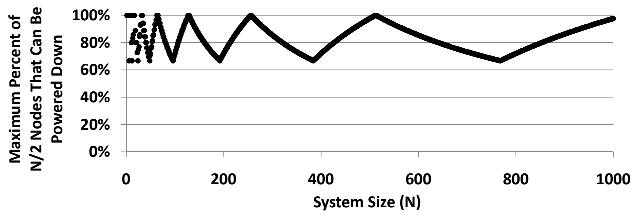
**Figure 4. Energy savings v/s varying system utilization.**



(a) Dissolving Chains



(b) Blinking Chains



(c) Dissolving Chain maximum number off offline nodes

**Figure 5. (a-b) Maximum node utilization as we iteratively take nodes offline. (c) Ability of Dissolving Chains to power down half of the nodes.**

### 5.3.2 *Database Scan Workload*

For the scan workload, increased utilization corresponds to increasing the length of time that an instance runs (to mimic what would happen if we turned nodes offline for such workloads). The results for this workload are presented in Figure 3. Again, the energy model is linear, but for scan the response time model is logarithmic. The average response time curve is sublinear as the pre-fetching used by the DBMS decreases the per-record response time as we increase the amount of data that is read. While the energy consumption curves in Figures 2 and 3 are both linear, as utilization increases, energy consumption grows faster with the CPU-bound index workload.

## 5.4 Effect of Decreasing Utilization

Using the models described in the previous section, we now apply the workload models to a $N = 1000$ system configuration under varied system utilization.

We then analyze the workload energy consumption of the overall system as the overall system utilization decreases from 100%. In addition to comparing differences between our methods, we also compare against the *Unmanaged* system, where all nodes are always online regardless of the overall system utilization.

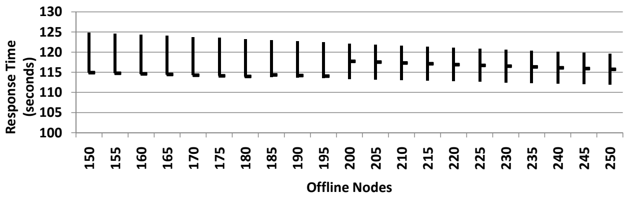
These results are shown in Figures 4 (a) and (b). In these figures, we vary the system utilization from 100% to 50% as shown on the x-axis (going from 100% on the left to 50% on the right). So going left to right, corresponds to decreasing the overall system utilization from the fully loaded (100%) system. For each point in these figures, we apply our empirically derived models from Section 5.3 to calculate the energy consumption. Using this calculated energy consumption, we plotted, on the y-axis, the energy saved by the entire system compared to the energy consumption at the 100% point.

We notice that an unmanaged cluster saves at most 10% in energy consumption (for the Index query workload Figure 4 (a)) at 50% utilization. For the Scan workload (Figure 4 (b)), the unmanaged cluster only saves 3% of energy at 50% utilization! However, using DC and BC, we can save 48% and 50% of the energy consumption at 50% utilization respectively. Notice, because of DC's inability to power down 500 nodes for $N = 1000$, its savings is slightly lower than BC.

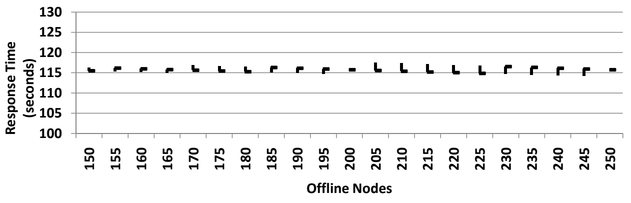
Another striking observation from Figure 4 is that the curves for both BC and DC have big swings/spikes. These spikes can be seen for both methods clearly in Figure 4 (b). This behavior is because both methods introduce load imbalances at certain operating states. Notice that the swings for BC are more gradual compared to DC – this is because BC maintains optimal load balance on the online nodes at any given operating state, which makes its energy swings are more subtle compared to DC.

Let us explore these swings in greater detail. Consider Figures 5 (a) and (b), where we power down $m$ nodes when the system utilization is $(1000 - m)/1000$ for a 1000 node system. As the system utilization drops, consider taking nodes offline one by one (incrementing $m$ by 1), up to a maximum of 500 nodes, using both DC and BC methods. Note that not all states will be balanced.

Figures 5 (a) and (b) show the *maximum node utilization* for both methods, i.e., the maximum relative increase (compared to $m = 0$) that any system node will see. Note the maximum node utilization is a crude way to determine the imbalance of the system. (This type of analysis can be used to avoid load spikes seen in Figure 1.) Comparing these two figures, we see that BC is more graceful in its worst-case



(a) Dissolving Chains Response Time



(b) Blinking Chains Response Time

**Figure 6. Comparing imbalanced operating points using the Index Query workload. The vertical lines in represent the range between the minimum and the maximum response times and the horizontal bar is the median response time.**

node utilization in imbalanced states (where maximum node utilization is greater than 100%) compared to DC.

In addition, from Sections 4.1 and 4.2.3 we know that BC has 16 balanced states (see Section 4.2.3) for $N = 1000$ while DC only has 4. (These correspond to a 100% maximum node utilization in Figures 5 (a) and (b).) Furthermore, even when both methods are imbalanced, BC has a better worst-case behavior than DC, as is evidenced by the lower height (node utilization) of the operating points in Figures 5 (a) and (b). For example, with respect to our problem statement in Section 2.2, if $U = 120\%$, then BC has 67 states where the maximum node utilization violates this constraint while DC has 209 states. This is simply a count of all possible operating states with a maximum nodes utilization greater than $U$.

Lastly, we notice from in Figure 5 (a) that DC cannot reach $\bar{s}(500)$ for $N = 1000$. This is because as it systematically traverses the ring, cutting segments in half, it may create irreducible segments of length 2. Thus, it cannot reach the optimal number of offline nodes. This effect can be seen in Figure 4, where near 50% utilization, DC is slightly lower in energy savings than BC.

An analysis of this phenomenon over varying system sizes ($N$) is shown in Figure 5 (c). Here we show how close DC can come to powering down $N/2$ nodes for $1 \leq N \leq 1000$. What we notice is that there are dramatic swings, but more importantly, we notice that DC can transition to $\bar{s}(N/2)$ only when $N = 2^i$. Ultimately, the reason this occurs is because DC heuristically takes nodes down and will never self-correct by bringing them back online as utilization monotonically decreases. The upside to this heuristic is a low (constant) transitioning energy cost that is discussed in Section 5.5.

For a detailed look at further effects of BC optimal load balancing to DC heuristic balancing, we zoom in on a smaller set of operating states. We use the models of Figures 2 and 3 and compare how energy consumption and response time are affected by these imbalanced states. Figure 6 examines the imbalanced operating points for the range of 150 to 250 offline nodes, in 5 node increments, while executing the Index query workload (Figure 2). (The results for the scan query workloads are similar and omitted here.) Figures 6 (a) and (b) compare the variance in node response time between the operating states for DC and BC, respectively. The response time variance is clearly far smaller with BC.

To summarize, BC transitioning results in more balanced node loads than DC. With its lower maximum node utilizations, BC offers greater opportunities to power down nodes and stay within the threshold $U$ in our problem statement (see Section 2.2).

| Methods | Properties: Load Balancing | Properties: Transitioning Overhead |
|---|---|---|
| Blinking Chains | Good | High |
| Dissolving Chains | Fair | Low |
| Mirroring | Poor | Low |

**Figure 7. Comparison of energy management methods**

## 5.5 Effect of Transitioning Costs

In the results above, we have not included any energy or latency costs associated with making transitions from one state to the next. From Section 5.4, we know that BC is optimal in balancing the load across the nodes, but the cost of this optimality is a complex transitioning mechanism (cf. Section 4.2.2). In contrast, DC always powers up/down the minimal number of nodes required to reach the target operating state. From the perspectives of energy consumed during the actual transitions, DC is clearly more efficient.

We have also examined the effect of the transitioning costs on BC. These results show that the BC transitions costs are acceptable if the system does not transition between states very often, and that for certain states DC will always be worse than BC. More details about this experiment can be found in [11]

## 5.6 Summary

Now we summarize some of the practical implications of our work. In a setting where load balance is not as important, as we discussed in Section 3.1, simple mirroring can be used. The power down scheme is simple (turn off one of the two replicate nodes, causing a 2X load increase on the remaining node) and it affords the 100% and 50% online balanced states. However, in cases where the huge 2X load imbalances must be avoided (in most cases involving SLAs), we suggest the Dissolving Chain (DC) and the Blinking Chain (BC) methods.

The differences between DC and BC are summarized in Figure 7. If avoiding load imbalances and the variation in loads across the nodes is important, then BC offers excellent load balancing in energy saving states. However, BC requires significant state transitioning overhead that would be amplified when system utilization is highly variable. Thus, if one knows the system utilization will be highly variable, DC offers low transitioning cost but incurs slight but predictable load imbalances and offers fewer state transitions.

Finally, notice that since both schemes leverage Chained Declustering, the usage of one over the other is not exclusive; if utilization fluctuates, we can switch to DC, and if there is little fluctuation, we can switch to BC. We will examine such hybrid approaches as part of future work.

## 6 Related Work

There has been considerable work on reducing the energy consumption of data centers, largely directed towards reducing the TCO [13], and include methods such as using more efficient power supply parts, raising data center temperatures, etc. These efforts are orthogonal to software methods for reducing energy consumption.

On the software systems side, a desired property is energy proportionality – i.e., an X% utilized server should consume X% of the peak 100% power. One of the hurdles in achieving this behavior is the problem that idle machines typically consume a significant amount (50%) of its peak power [3]. For web servers, methods such as [16, 18] propose selectively turning servers off. Additional methods [15, 17] either rely on learning request skew, specialized hardware, and data migration and do not explore load imbalances caused by powering down disks.

Another mechanism for energy management is to use VM-based consolidation methods, such as [2, 8, 19, 21, 22]. However these methods are challenging to use for data-intensive applications as they may require moving large amounts of data during VM migration.

Recent work by Leverich powers down MapReduce cluster nodes but does not consider load balancing [12].

None of these previous works have considered the problem that we address – namely, energy management using replication to maintain data availability, while maintaining a well-balanced system.

## 7 Conclusions and Future Work

In this paper we have presented energy management methods that can be used in distributed data processing environments to reduce energy consumption. We leverage the properties of replication schemes and design techniques that can take nodes offline to conserve energy when the system utilization is low. Our methods trade off load balancing against energy efficient state transitioning, allowing the user to choose a suitable strategy. To the best of our knowledge, this is the first paper that makes a connection between replication, energy management and load balancing.

This paper seeds a number of directions for future work. First, our methods used a 2X replication, and does not exploit utilization below 50%. One direction for future work is to build on the ideas proposed here and broaden the connections between generic levels of replication, load balancing and energy management. Other directions for future work include incorporating workload modeling and prediction techniques to work with our method, techniques that switch between Blinking and Dissolving Chains based on hybrid workload characteristics, and improving the techniques for handling rapid transitions between different operating states.

Finally, we fully recognize that replication, power down sequences, and load balancing are only part of a larger software solution for energy management in data intensive computing environments. We recognize that extensions to our work are needed to produce fully deployable complete solutions (e.g. incorporating workload modeling), and it is our hope that this work instigates other work in this emerging area of research.

## 8 References

[1] Power and Temperature Measurement Setup Guide SPECpower V1.1. *SPEC Power*, 2010.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In *SOSP*, 2003.

[3] L. A. Barroso and U. Hölzle. The Case for Energy-Proportional Computing. *IEEE Computer*, 40(12), 2007.

[4] C. Belady. In the Data Center, Power and Cooling Costs More than the IT Equipment it Supports. *Electronics Cooling*, 23(1), 2007.

[5] D. Bitton and J. Gray. Disk Shadowing. In *VLDB*, 1988.

[6] A. Borr. Transaction Monitoring in Encompass. In *VLDB*, 1981.

[7] K. G. Brill. Data Center Energy Efficiency and Productivity. In *The Uptime Institute - White Paper*, 2007.

[8] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live Migration of Virtual Machines. In *NSDI*, 2005.

[9] D. J. DeWitt. The Wisconsin Benchmark: Past, Present, and Future. In J. Gray, editor, *The Benchmark Handbook for Database and Transaction Systems (2nd Edition)*. Morgan Kaufmann, 1993.

[10] H.-I. Hsiao and D. J. DeWitt. Chained Declustering: A New Availability Strategy for Multiprocessor Database Machines. In *ICDE*, 1990.

[11] W. Lang, J. M. Patel, and J. F. Naughton. On Energy Management, Load Balancing and Replication. Technical Report UW-CS-TR-1670, University of Wisconsin–Madison; Computer Sciences Department, 2010.

[12] J. Leverich and C. Kozyrakis. On the Energy (In)efficiency of Hadoop Clusters. In *HotPower*, 2009.

[13] C. D. Patel and A. J. Shah. Cost Model for Planning, Development and Operation of a Datacenter. *HP Technical Report, HPL-2005-107R1*, 2005.

[14] D. A. Patterson, G. Gibson, and R. H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *SIGMOD*, 1988.

[15] E. Pinheiro and R. Bianchini. Energy Conservation Techniques for Disk Array-Based Servers. In *ICS*, 2004.

[16] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath. Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems. In *Workshop on Compilers and Operating Systems for Low Power*, 2001.

[17] E. Pinheiro, R. Bianchini, and C. Dubnicki. Exploiting Redundancy to Conserve Energy in Storage Systems. In *SIGMETRICS*, 2006.

[18] K. Rajamani and C. Lefurgy. On Evaluating Request-Distribution Schemes for Saving Energy in Server Clusters. In *Proc. of the IEEE Intl. Symp. on Performance Analysis of Systems and Software*, 2003.

[19] P. Ranganathan, P. Leech, D. Irwin, and J. Chase. Ensemble-level Power Management for Dense Blade Servers. In *ISCA*, 2006.

[20] Teradata. DBC/1012 Database Computer System Manual Release 2.0. *Technical Document C10-0001-02*, 1985.

[21] N. Tolia, Z. Wang, M. Marwah, C. Bash, P. Ranganathan, and X. Zhu. Delivering Energy Proportionality with Non Energy-Proportional Systems - Optimizing the Ensemble. In *HotPower*, 2008.

[22] C. A. Waldspurger. Memory Resource Management in VMware ESX Server. In *OSDI*, 2002.

# Third Int'l Workshop on "Personalized Access, Profile Management, and Context Awareness in Databases" (PersDB 2009)

Sihem Amer-Yahia
Yahoo! Research New York
111 W 40th Street, New York
NY, USA
sihem@yahoo-inc.com

Georgia Koutrika
Computer Science Dept., Stanford University
353 Serra Mall, Stanford
CA, USA
koutrika@stanford.edu

## 1. INTRODUCTION

Proliferation of database-driven web sites has brought upon a plethora of information access and dissemination applications. Monitoring and trading stock portfolios, news notification, weather tracking, and even simple search are just a few examples. In addition, emerging applications such as Web-based communities, wikis, social networks, mashups and folksonomies enhance creativity, information sharing, and collaboration among users providing richer interaction possibilities. Now, users can not only access content but they can also generate, share and modify content (both theirs and others'), compose their applications, enhance their interface, etc.

In all these applications, different notions of user information, such as preferences, community memberships and social interactions, and context information, such as a user's social network, location, time, and other features of a user's environment, are of paramount importance in order to improve and personalize user experience. In this context, new challenges emerge for user-centric, context-aware database systems for storing and managing different aspects of user and context information, for data management and computing taking into account personal, social and contextual information about users and for customizability of their behavior.

There are several research efforts that are trying to address research problems that arise when developing such systems. For example, user preferences may be associated with the search process and interpreted as preferences over the desired system answers (e.g., [3, 4]) or they may be associated with aspects of the data management process specifying user requirements for response times, data freshness, quality of service etc, and used in mechanisms, such as caching, query admission control, resource allocation, and scheduling (e.g., [2, 5]). Likewise, a user's context can be the social network or community where a user belongs. A user may belong to more than one network and community. Identification of a user' community is important as well as building personalized recommendations, user collaboration and sharing schemes that address communities [1, 6].

The purpose of the PersDB Workshop ("Personalized Access, Profile Management, and Context Awareness in Databases") is to bring together researchers and practitioners both from the academia and the industry and provide a forum for presentation of the latest research results, new technology developments, and new applications in the areas of personalized access, profile management, and context awareness in database systems.

The $3^{rd}$ Int'l Workshop on "Personalized Access, Profile Management, and Context Awareness in Databases" (PersDB 2009) was held in conjunction with VLDB in Lyon, France. We have received 18 papers spanning different topics including context-driven databases, recommendations, search and social networks. We accepted 6 papers, having a healthy acceptance rate of about 33%.

## 2. WORKSHOP OUTLINE

### 2.1 Invited Talk

Recommender systems have been extremely successful in reaching relevant content to users and they are very popular on sites, such as Amazon, Netflix and Google News. These systems incorporate endorsements of items by other users and/or ratings provided by the same user and they recommend items to the user that are likely to be of interest. *Laks V.S. Lakshmanan* (University of British Columbia, Canada), who gave the keynote talk titled "*Recommender Systems Revisited: from Items to Transactions*", described a new idea: developing recommendation strategies and systems not just for recommending items but for users performing transactions.

The invited speaker described the notion of an *exchange market* (e.g., *ReadItSwapIt.co.uk*), a social network where users register items (e.g., toaster, lawn mower) they are willing to give away to other users in exchange for items in their wish list which they have registered with the system. Users either swap items or more generally exchange items in cycles. He described approximations as well as heuristic algorithms for recommending exchange transactions to users assuming relatively simple transaction cycles.

Recommending transactions where items are swapped is a new idea which raises many challenging problems that go beyond item recommendations and opens several interesting directions for future work. For instance, one challenge is to consider more complicated transactions involving several interesting parties that need to come together in order to achieve a transaction. Another interesting direction is to consider exchange markets that change over time and model different possible objectives that may characterize transactions, such as fairness and average waiting time.

### 2.2 Panel: "How Far Should We Personalize?"

Given the proliferation of data and applications, different possibilities as well as different requirements for personalizing user experience emerge at various levels (e.g., content, UI, services, etc). For instance, people may like different services on their cell phone or in Facebook. They may be interested in different content depending on their location, task, preferences or group they belong to. Different presentation features serve different people better. For example, some users like lengthy explanations, others may want to see reviews. Endless personalization possibilities seem to exist in different applications, from personalized search and ads to personal mashups. At the same time, there are several arguments against (over-) personalization.

For example, if we do not have correct information about a user, personalization may hurt accuracy. In addition to gathering and maintaining a profile, on-the-fly personalization can be expensive. Over-personalization may lead to over-specialization. Making a recommendation of something a user would definitely like is valuable but what about serendipity and diversity? In advertising, there is also a delicate balance between the accuracy of ads and their total irrelevance. From the publishers point of view, they would rather serve relevant content than not, from the users' perspective, more relevant ads may be annoying and more intrusive because they may exploit sensitive, personal, information (e.g., a user announces the birth of a child in an email to friends, ads start suggesting where to buy baby stuff).

With all these in mind, the workshop panel's subject was "How Far Should We Personalize?" and its objective was to discuss when, what, how, and to what extent we should or should not personalize. Five panelists coming both from academia and industry shed light on the subject from different perspectives.

*Sihem Amer-Yahia* (Y! Labs, USA) pointed out that personalization is helpful if we have enough information about the user and the topic. Unfortunately, most users are tail users and there is little information about all possible topics. Indicatively, she mentioned that in delicious, only 3.5% of the URLs are tagged by more than 5 users. Given the fact that we may not have enough information for all topics and all users, personalization needs to be performed in a granular fashion: For head users/topics, *personalization* is suitable. For lesser known users/topics, *groupalization* at various levels is more suitable. Finally, for unknown users/topics, we need to resort back to traditional methods with no personalization.

On the other hand, *Yannis Ioannidis* (University of Athens, Greece) argued that in many scenarios we have various different sources of information for a user, including explicit statements, such as ratings, implicit feedback, such as selecting a product, and social activities, such as blogging. In addition, a user connects to other people in many ways and these connections can possibly lead to learning more about a particular user. Hence, in these cases, the problem is more complex. First, we need to understand the meaning and contribution of each possible piece of information that may be explicitly or implicitly related to a user, and then we need appropriate *methods to extract, detect, propagate, derive and synthesize user information*.

*Christian Jensen* (Aalborg University, Denmark) described the importance of *geo-context awareness* for the mobile Internet and for service customization. Mobile phones are now providing GPS functionality. An emerging standard is also implemented in Firefox to enable sites to ask the browser for the user's location, which includes current location, destinations visited, routes followed, and so forth. At the same time, there is the "dark side" of personalization: For example, knowing someone's location may be misused in a number of ways. There is a need to develop privacy-preserving techniques that offer (useful) tradeoffs among the degree of privacy, the cost or performance and the quality of service. In addition, it is important to offer transparency, i.e., enable users to understand the privacy proposition and opt-in in a flexible and fine-grained way. In brief, we need to enable *privacy-aware personalization*. *Elisa Quintarelli* (Politecnico di Milano, Italy) also discussed the notion of context in personalization, where the context can also include other information apart from location, such as user roles, interests, and temporal information.

Finally, *Evi Pitoura* (University of Ioannina, Greece) pointed out that personalization is not just about optimizing relevance for a particular user. Other factors must be considered, such as *diversity* and building *trust*. In addition, it is easier to judge positively a search result/recommendation/query ranking when there is some explanation of its derivation. On the other hand, *over-personalization* may be annoying: too many personal options, too many personal ads may just overwhelm and drive away the user.

## 2.3 Paper Presentations

### 2.3.1 Collaborative Efforts

*Vagelis Hristidis* and *Eduardo Ruiz* in their paper "*CADS: a Collaborative Adaptive Data Sharing Platform*" describe the challenges and initial design ideas for building a collaborative adaptive data sharing platform, which facilitates data annotation at insertion-time and leverages these annotations at query-time. CADS learns with time the information demand (query workload), which is then used to create adaptive insertion and query forms. The goal of CADS is to allow the effortless sharing of documents, while at the same time serving semi-structured queries.

*Giorgos Giannopoulos, Theodore Dalamagas* and *Timos Sellis* in their work "*Collaborative Ranking Function Training for Web Search Personalization*" present a framework for improving the ranking function training and the re-ranking process for web search personalization. They utilize clickthrough data from several users in order to create multiple ranking functions, each one corresponding to different topic areas. Those ranking functions are combined each time a user poses a new query in order to produce a new ranking, taking into account the similarity of the query with each of the topic areas.

### 2.3.2 Personalization

*Alessandro Campi, Mirjana Mazuran* and *Stefania Ronchi* in their paper titled "*Domain Level Personalization Technique*" learn user preferences by tracking user choices on search results that are presented in clusters. They apply user preferences to (a) rank the clusters of a search result list, showing in the first positions the clusters containing more interesting contents from the user viewpoint, (b) rank the documents contained in each cluster accordingly, and (c) recommend a set of terms taken from the user's profile for query expansion.

*Huiming Qu, Jie Xu* and *Alexandros Labrinidis* in their paper titled "*Guiding Personal Choices in a Quality Contracts Driven Query Economy*" deal with user preferences regarding quality-of service (QoS) and quality-of-data (QoD). Given an environment where user preferences over different quality dimensions are expressed using quality contracts that are attached to queries, they look at how these contracts can be adapted over time. They propose an adaptive strategy, which monitors a user's queries and the server's responses and automatically adapts the quality contracts of subsequent user-submitted queries.

### 2.3.3 Recommendations

*Sofiane Abbar, Mokrane Bouzeghoub* and *Lopes Stephane* in their paper "*Context-Aware Recommender Systems: a Service-Oriented Approach*" propose a context-aware recommender system. The proposed system consists of several personalization services that help introduce the notion of context in recommendations.

Finally, *Kostas Stefanidis, Marina Drosou* and *Evi Pitoura* in their paper "*You May Also Like Results in Relational Databases*" consider recommending to the users of a database not only tuples in the results of their queries but additional tuples that may be of potential interest. They propose three approaches to compute such results that exploit: (a) the content and schema of the current query result and database instance, or (b) the history of previously submitted queries to the database system, e.g. by using query logs, or (c) resources external to the database, such as related published results and reports.

## 3. WORKSHOP CONCLUSIONS

Given the proliferation of data and applications, different possibilities for personalizing user experience as well as for adapting aspects of the data management process, such as response times, data freshness and quality of service, emerge. The wide use of advanced mobile devices opens up the way to new opportunities for personalized services that take into account not just user preferences but also the user context, such as location and situation.

Obviously, we need to see personalization at various levels and from different aspects. We need to offer personalization in a granular level by dynamically taking into account how much we know about a user or a topic to determine how far we should go. One emerging trend is leveraging the collaborative efforts and the wisdom of the crowds to better serve the needs of different user communities. We can also take advantage of the users' social activities, such as resource sharing and connecting, to derive information about individual users or groups of users and offer targeted services and content. How we can achieve all these is an open challenge.

Finally, in personalization, we need to consider several factors not just improving relevance. For example, we need to offer diverse results and we need to build trusted services. Privacy is a highly overlooked issue. Users need to understand and be able to control how personalization interferes with their privacy. To this end, we need to build privacy-aware personalization.

## 4. REFERENCES

[1] S. Amer-Yahia, S. Basu Roy, A. Chawla, G. Das, and C. Yu. Group recommendation: Semantics and efficiency. In *VLDB Conference*, 2009.

[2] M. Cherniack, E. Galvez, M. Franklin, and S. Zdonik. Profile-driven cache management. In *ICDE Conference*, 2003.

[3] W. Kießling and W. Kostler. Foundations of preferences in database systems. In *VLDB Conference*, 2002.

[4] G. Koutrika and Y. Ioannidis. Personalized queries under a generalized preference model. In *ICDE Conference*, 2005.

[5] H. Qu and A. Labrinidis. Preference-aware query and update scheduling in web-databases. In *ICDE Conference*, 2007.

[6] Y. Zhou, G. Cong, B. Cui, C. Jensen, and J. Yao. Routing questions to the right users in online communities. In *ICDE Conference*, 2007.

# Second Workshop on Very Large Digital Libraries

**In conjunction with the European Conference on Digital Libraries**
**Corfu, Greece, 2 October 2009**

Paolo Manghi
ISTI - Consiglio Nazionale delle Ricerche
Pisa, Italy
paolo.manghi@isti.cnr.it

Pasquale Pagano
ISTI - Consiglio Nazionale delle Ricerche
Pisa, Italy
pasquale.pagano@isti.cnr.it

Yannis Ioannidis
University of Athens
Athens, Greece
yannis@di.uoa.gr

## 1. MOTIVATIONS

The mission of the international workshop series on Very Large Digital Libraries (VLDLs) is to provide researchers, practitioners and application developers with a forum fostering a constructive exchange among all key actors in the field of Very Large Digital Libraries. Its long-term and ambitious goal is to discuss the foundations of VLDLs and establish it as a research field on its own, with well-defined areas, models, trends, open problems, and technology.

The main outcome of the first VLDL workshop [1] was consolidation of its mission. After several presentations on the papers accepted to the workshop and further demonstrations and long debates by all participants, there was eventual agreement that VLDLs deserve a chapter of their own in computer science research. Experience in the field since the historical beginning of DLs has proven that VLDLs cannot be simply regarded as very large databases storing Digital Library content, as one may be tempted to assess. In fact, as the DELOS Reference Model for Digital Libraries [4] well motivates, DL Systems cannot be approached from the perspective of content management only; the dimensions of user, functionality, policy, quality, and architecture management are equally important. Accordingly, DLs become Very Large DLs (VLDLs) when any one of these aspects reaches a magnitutde that requires specialized technologies.

VLDLs are clearly in their early stage of development. For example, interdependencies among the aspects above are still to be identified and studied; the same holds for models and measures for evaluating "very-largeness" of given DL systems. In fact, it is not even clear if there are clear enough patterns and best practices that are common in existing DL systems to determine the boundaries of the field or, instead, practical experience is still too much in its infancy.

The second VLDL workshop continued such investigations from where the first one left off. To this aim, its call for papers focused on foundational aspects of VLDLs and real experiences with them:

**Foundational topics** They covered definitional models and measures (content, functionality, users, and policies), architectural models, and design methodologies for VLDLs.

**System topics** They covered ideas, experiments, and practical experiences in system design and implementation. Of particular interests were the following: integration and federation of DLs, user management, security, sustainability, scalability, distribution, interoperability for content and functionalities, quality of service, storage & indexes, and preservation.

All contributions submitted were peer reviewed by two of the six members of the Program Committee and nine were accepted. The workshop structure comprised an invited speakers session followed by the presentation of the nine contributions, organized into three sessions: *systems*, *data management* and *functionalities* for VLDLs. Each session is analyzed in a separate subsection below.

## 2. WORKSHOP PRESENTATIONS

### 2.1 Invited talks

This session was dedicated to present the experiences of design and development of two major projects, funded by the European Commission, whose challenges are very large in terms of content heterogeneity and size:

*Building Europeana v1.0: Towards a Large-Scale Content Ingestion*: Julie Verleyen (Europena Office, Koninklijke Bibliotheek, National Library of the Netherlands) presented the data ingestion workflow adopted in Europeana [5], based on the Open Archival Information System (OAIS) [3]. The workflow safely rules the flow of data from content providers to the final receiver, i.e. , Europeana, by addressing the functional needs of updates, traceability, duplication, and conversion.

*SAPIR: towards Large Scale Multimedia Content Search*: Fausto Rabitti presented the advanced solutions proposed by the SAPIR project [6] (coordinated by Maristella Agosti and himself) to enable content-based searches of audio-visual information in the presence of very large collections of images and audio pieces. The solution exploits a P2P-based architecture for feature extraction, designed to scale and process very large collections of digital objects.

### 2.2 Session on VLDL systems

This session focused on the problems arising in hardware and software architectures of very large digital library systems.

*Utility-based High Performance Digital Library Systems (by Hussein Suleman)*: Hussein Suleman presented an analysis of current high-performance systems and argued for the adoption of utility computing to tackle scalability issues that arise when dealing with large data collections under high quality-of-service requirements.

*MultiMatch: Multiple Access to Cultural Heritage (by Giuseppe Amato, Franca Debole, Carol Peters, and Pasquale Savino*: Franca Debole presented the main issues that surfaced while designing the MultiMatch system, enabling users to run searches over cultural heritage material across different media types and languages.

*Integrating Multi-Dimensional Information Spaces (by Kostas Saidis and Alex Delis*: Kostas Saidis defined very large digital libraries as those managing several large information spaces, not only in terms of data volume but also in terms of diversity of material and heterogeneity of sources they can support. Based on this, he presented a data management specification for digital libraries, enabling the construction of infrastructures for information space integration and interoperability.

## 2.3 Session on Data management in VLDLs

This session discussed experiences with dealing with different aspects of content management in the context of very large scenarios.

*Improving Similarity Search in Face-Images Data (by Pedro Chambel and Fernanda Barbosa)*: Finding all faces in a large data set that are similar to a given desired face is computationally very expensive. Fernanda Barbosa proposed an approach to face recognition based on metric spaces, in which images are mapped to metric data structures in a metric space and similarity searches are reduced to Euclidean-distance range queries within that space.

*Improving Query Results with Automatic Duplicate Detection (by Irina Astrova*: A significant challenge in ontology-based data integration is that the process of data integration may lead to duplicate attributes, i.e., attributes that appear different but whose semantics is, in fact, the same. Irina Astrova presented a context-based approach for automatic duplicate detection in very large ontologies.

*Enabling Content-Based Image Retrieval in Very Large Digital Libraries (by Paolo Bolettieri, Andrea Esuli, Fabrizio Falchi, Claudio Lucchese, Raffaele Perego, and Fausto Rabitti)*: The problems arising in the context of efficient Content-Based Image Retrieval (CBIR) are mainly concerned with the design of effective feature extraction algorithms. However, when the number of the images and/or the sizes of the individual images are very large, processing these images and building the corresponding indexes becomes prohibitively expensive. Fausto Rabitti presented the CoPhIR [9] collection (the largest available to the research community) of 106 million images and the GRID-based image crawling process used to extract the features required for CBIR.

## 2.4 Session on Functionality for VLDLs

This session attracted contributions related to functionality design and development affected by very large size.

*Semantic Journal Mapping for Search Visualization in a Large Scale Article Digital Library (by Glen Newton, Alison Callahan, and Michel Dumontier*: Glen Newton analyzed the scalability and utility of semantic mapping of journals in a large science, technology, and medical digital library, numbering 5,7+ Millions articles. The aim was to evaluate the effectiveness of semantic mappings of articles for query result refinement and visual contextualization in very large digital libraries.

*Exploiting Individual Users and User Groups Interaction Features: Methodology and Infrastructure Design (by Emanuele Di Buccio and Massimo Melucci*: Information gathered by monitoring interactions between users (or group of related users) and a DL system can be used as an indicator of individual user interests. Emanuele Di Buccio proposed a methodology based on a Peer-To-Peer infrastructure for collecting and exploiting behaviors of both individual users and user groups as a source of evidence of their interests.

*Maintaining Object Authenticity in Very Large Digital Libraries (by Tobias Blanke, Stephen Grace, Mark Hedges, Gareth Knight, and Shrija Rajbhandari*: DLs are increasingly used to manage large volumes of research data. Such data is irreplaceable and their management calls for scalable methodologies for its long-term access and curation. Mark Hedges presented the case study of an iRODS-based architecture [8] for automatic support of rule-based authenticity models in very large digital libraries.

## 3. WORKSHOP DISCUSSION

The final, brainstorming session of the workshop started by everyone agreeing that "very large" issues in Digital Libraries are concerned with four axes: user management, content management, functionality management, and policy management. For example, a Digital Library may be "very large" in terms of the amount of data it has to manage as well as in terms of the number of communities of end users it has to serve at the same time.

The discussion focused largely on *scalability*. In the database world, a database is very large when it can scale up beyond a given threshold of content (e.g., greater than 1 Terabyte of data). Can we draw a parallel between DLs and databases and claim that a DL is very large when it can scale up beyond a certain size in the four axes mentioned above? In other words, are DLs very large because of well-established *size* thresholds, be them related to users, policies, content, or functionalities? Furthermore, what are these thresholds and how are they to be identified? Analysis of known VLDL scenarios, e.g., those of e-Infrastructures such as Europeana and DRIVER, seemed to answer this question in the negative: DL scalability appears to depend on size as well as on at least two other important interrelated aspects: *sustainability* and *interoperability*.

In the DL world, scalability cannot be measured in an absolute sense as in the database world, where size determines whether or not a database is labeled as very large. In fact, the adjective “very large” strongly depends on sustainability issues, often ruling the DL world, where funds are generally scarce and hard to guarantee in depth of time. As a consequence, the adjective “very large” may label systems where the problem to be tackled might not look “that large” in other domains. For example, adoption of GRID infrastructures or high performance computing solutions used by the physics community to tackle very large datasets are generally not a reasonable solution in a typical DL application scenario, given the current dominant business models. As a consequence, new VLDL research goals arise, such as developing sustainable and low-cost hardware and software infrastructures or devising alternative business models.

The second key system property for VLDLs is interoperability with respect to content and functionality – whose foundations are the subject of several intense studies recently, e.g., in the context of the DL.org project [7]. For example, Europeana and DRIVER infrastructures aggregate and integrate tens of millions of metadata records from different data sources. This volume of data, however, does not introduce scalability issues, as it can be effectively dealt with standard storage and indexing technology. To the contrary, such systems are not very large in terms of content management, but rather in terms of the arbitrary amount of distributed data sources they have to federate and the heterogeneity of technology, data structure, and data semantics they have to cope with to accomplish this task. Similar reasoning can be applied to other categories of content to be integrated, such as ontologies, policies, user profiles, and others, where diversity of information representation and forms of information exchange are the real “very large” issues.

## 4. CONCLUSIONS

The main conclusion drawn from all workshop deliberations was that Very Large Digital Libraries research seem to focus on scalability challenges that Digital Library systems manifest in terms of content, user, policy, and functionality management. Furthermore, it is not only size that is a key issue but also levels of sustainability and interoperability. The need for research in these areas is great, and any progress in these directions will be further investigated in the next edition of VLDL.

## 5. ACKNOWLEDGMENTS

We would like to thank all those who contributed directly or indirectly to the event, especially our colleagues at ISTI-CNR Donatella Castelli, Leonardo Candela, and Costantino Thanos for their research inspiration, and Vittore Casarosa and Francesca Borri for the logistic support.

Special thanks are also due to the members of the program committee, *Stavros Christodoulakis* (Technical University of Crete (MUSIC/TUC), Greece) *Stefan Gradman* (Institut fr Bibliotheks und Informationswissenschaft, Humboldt-Universitt zu Berlin, Germany), *Kat Hagedorn* (OAIster System, University of Michigan Digital Library Production Service, USA), *Dean B. Krafft* (National Science Digital Library Project, Cornell Information Science, USA), Yosi Mass (IBM Research Division, Haifa Research Laboratory, University Campus, Haifa, Israel), *Peter Wittenburg* (Max-Planck-Institute for Psycholinguistics, The Netherlands), whose long research experience contributed in making this workshop an attractive and fruitful experience for all authors and participants.

Finally, our sincere gratitude goes to the chairs of the sessions, who offered their expertise in coordinatin and harmonizing the sequence of presentations, and of course, to all authors, whose passion and ideas were the real fuel of VLDL 2009.

Workshop proceedings [2] were printed by the DELOS Association for Digital Libraries [4].

## 6. REFERENCES

[1] Paolo Manghi, Pasquale Pagano and Pavel Zezula. Proceedings of the First Workshop on Very Large Digital Libraries, held in conjunction with ECDL 2008, Aarhus, Denmark, 2008

[2] Yannis Ioannidis, Paolo Manghi, Pasquale Pagano. Proceedings of the Second Workshop on Very Large Digital Libraries, held in conjunction with ECDL 2009, Corfu, Greece, 2009

[3] OAIS: Open Archival Information System. http://public.ccsds.org/publications/archive/650x0b1.pdf

[4] DELOS: Digital library rEference modeL and interOperability Standards. http://www.delos.info

[5] Europeana: Connecting cultural heritage. http://www.europeana.eu

[6] SAPIR: Search In Audio Visual Content Using Peer-to-peer IR . http://www.sapir.eu

[7] DL.org Project: http://www.dlorg.eu

[8] iRods Project: Integrated Rule-Oriented Data System http://www.irods.org

[9] CoPhIR Image Collection: Content-based Photo Image Retrieval Test-Collection http://cophir.isti.cnr.it

# 5th International Workshop on Networking Meets Databases (NetDB 2009)

Boon Thau Loo
University of Pennsylvania
3330 Walnut Street, Philadelphia, PA 19104
boonloo@cis.upenn.edu

Stefan Saroiu
Microsoft Research
One Microsoft Way, Redmond, WA 98052
ssaroiu@microsoft.com

## 1. INTRODUCTION

The Workshop on Networking Meets Databases (NetDB) is a venue that aims to bring together researchers from the systems and networking community and the database community. Many current research areas, such as cloud computing, data-center networking, sensor networks, network management, or social networks, raise research problems that lie at the boundary between these two communities. The workshop's goal is to foster an environment in which researchers from both communities can discuss ideas that will shape and influence these emerging research areas. The workshop encourages submissions of early work, with novel and interesting ideas. The expectation is that work introduced at NetDB, once fully thought through, completed, and described in a finished form, may be relevant to conferences such as SOSP, OSDI, SIGCOMM, SIGMOD, VLDB, NSDI, or ICDE.

Traditionally, NetDB has been co-located with either the IEEE International Conference on Data Engineering (ICDE'05, ICDE'06, ICDE'08) or the USENIX Symposium on Networked Systems Design and Implementation (NSDI'08). This year is the first time when the workshop was co-located with the ACM Symposium on Operating Systems Principles (SOSP). By co-locating with SOSP, we aimed at raising the interest from the wider systems community on topics relevant to NetDB.

## 2. REVIEW PROCESS

NetDB received a total of 16 submissions and many of these submissions were of very high quality. Most papers covered topics in hot and emerging research areas relevant to NetDB, such as declarative systems, online social networks, and large-scale MapReduce-like systems. As in previous years, NetDB continued to attract novel work at the confluence of the networking and database communities. All papers received three reviews, and we added a fourth review for borderline cases. We held a phone meeting that lasted an hour and a half. During the meeting, we discussed 14 papers and we accepted seven.

## 3. WORKSHOP DAY

NetDB 2009 was held on the final day of SOSP (October 14th, 2009) starting at 1:30pm. The workshop program included seven paper presentations, followed by a 1.5 hour panel discussion. While we intended the workshop to conclude by 7pm, the panel fostered a healthy debate that made us end the workshop closer to 8pm. Officially, NetDB had 21 registrations, although we counted more people in the audience. More details of the program can be found online on the workshop's website.

### 3.1 Presentations

The accepted papers covered topics in cloud computing (one paper), social networks (one paper), publish-subscribe systems (one paper), RFID monitoring (one paper), and declarative systems (three papers). The papers along with the slides used in their presentations can be found on the workshop's website. The accepted papers were:

- **Data Indexing for Stateful, Large-scale Data Processing**. Dionysios Logothetis and Kenneth Yocum (UC San Diego).
  **One-line summary:** This paper presents techniques to integrate indexes with stateful/incremental batch processing at large scale (systems such as MapReduce restart the complete computation when presented with data).

- **Scaling Online Social Networks without Pains.** Josep M. Pujol, Geogos Siganos, Vijay Erramilli, Pablo Rodriguez (Telefonica Research).
  **One-line summary:** This paper describes a mechanism for partitioning a graph among multiple servers, so that social network computations on the graph can be carried out in parallel.

- **Generating Wide-Area Content-Based Publish/Subscribe Workloads.** Albert Yu, Pankaj K. Agarwal, Jun Yang (Duke University).
  **One-line summary:**This paper describes a workload generator for publish/subscribe systems parameterized by data gathered from Google Groups.

- **Architectural Considerations for Distributed RFID Tracking and Monitoring.** Zhao Cao, Yanlei Diao, Prashant Shenoy (University of Massachusetts).
  **One-line summary:** This paper considers centralized and distributed architecture designs for RFID monitoring systems that combine inference and query processing techniques.

- **Declarative Transport: A Customizable Transport Service for the Future Internet.** Karim Mattar, Ibrahim Matta, John Day, Vatche Ishakian, Gonca Gursun (Boston University).

**One-line summary:** This paper uses declarative networking techniques to specify and implement policies used in transport protocols.

- **I Do Declare: Consensus in a Logic Language.** Peter Alvaro, Neil Conway, Russell Sears, Tyson Condie, Joseph M. Hellerstein (UC Berkeley).
  **One-line summary:** This paper uses declarative networking techniques to implement the Paxos consensus protocol, and shows that primitives used in consensus protocol specifications map directly to simple declarative networking language constructs.

- **On the Declarativity of Declarative Networking.** Yun Mao (AT&T Labs - Research).
  **One-line summary:** This paper surveys recent systems that are based on the use declarative networking, in order to investigate their level of declarativity.

Each paper was allocated a 30 minute slot that included 5-7 minutes allocated to Q&A.

### 3.2 Panel Discussion

During the final hour and a half of the workshop, we held a panel titled "Declare your declarativity". The panelists are Fred Baker (Cisco), Joe Hellerstein (UC Berkeley), Eddie Kohler (UCLA), Arvind Krishnamurthy (University of Washington), Petros Maniatis (Intel Research Berkeley), and Timothy Roscoe (ETH Zurich). We also were able to have a small wine and beer open bar that we think helped ignite the panel discussion.

The purpose of the panel was spurred by the recent interests in the use of declarative languages to implement complex systems and networking protocols. No less than three papers accepted at NetDB 2009 covered topics in declarative systems. The appeal of declarative languages is their compactness: with just a few tens of lines, one can describe the entire semantics of a complex protocol, such as the consistency semantics of a file-system, or the TCP protocol, or the Paxos consensus protocol. Such compactness can make it easier for developers to ensure that their protocol implementations are semantically correct and they have few bugs. At the same time, there is little experience with using declarative languages for solving systems problems, and some are skeptical about their ease of use and ease of debugging.

Each member of the panel took a stand on whether systems will benefit from using declarative languages. The discussion raised a couple of interesting points:

- An argument in favor of declarative systems is that declarative implementations are brief (few number of lines of code). Some people pointed out that lines of code is not an adequate metric for measuring how good an implementation is.

- Another benefit of declarative systems is that it gives semantics to an implementation and it *could* make it easier to build systems on top of these implementations. For example, it might make it easier to run verification tools or bug findings tools. The consensus was that this might be a fruitful future direction for this research area.

- A system might be more successful when written in programming languages everyone understands.

- Many of the declarative systems implemented by the research community already have robust implementations running in industry. Instead, implementing declarative systems that industry needs might have a bigger impact.

- Any system has a configuration. Many configuration files today are already written in a declarative fashion.

- Some rule-of-thumbs used when building systems appear to be hard to capture declaratively. For example, "be liberal in what you receive and conservative in what you send" is a principle for building networked systems. It is unclear whether a declarative implementation can capture such a principle.

This brief list is far from a rigorous summary of the panelists' points of view. Instead, we encourage the reader to visit the NetDB's website where the panelists' slide decks are posted.

## 4. WORKSHOP INFORMATION

### 4.1 Program Committee

The program committee of NetDB 2009 was formed by:

- Atul Adya, Google
- Brian Cooper, Yahoo! Research
- Mary Fernández, AT&T Labs
- James Hamilton, Amazon,
- Joe Hellerstein, UC Berkeley
- Boon Thau Loo, University of Pennsylvania (co-chair)
- Sam Madden, MIT
- Ratul Mahajan, Microsoft Research
- Petros Maniatis, Intel Berkeley
- Stefan Saroiu, MSR Redmond (co-chair)
- Emin Gün Sirer, Cornell

### 4.2 Workshop Website

`http://netdb09.cis.upenn.edu`

### 4.3 Acknowledgments

We would like to thank Marc Fiuczynski, the SOSP workshop chair, for his help with the workshop organization. We are also very grateful to Microsoft Research Redmond for a generous sponsorship.

# Call for Participation
# First ACM Symposium on Cloud Computing (SOCC)
**June 10 & 11, 2010, Indianapolis**
**http://research.microsoft.com/socc2010**

The *ACM Symposium on Cloud Computing* 2010 (*ACM SOCC 2010*) is the first in a new series of symposia with the aim of bringing together researchers, developers, users, and practitioners interested in cloud computing. This series is co-sponsored by the ACM Special Interest Groups on Management of Data (ACM SIGMOD) and on Operating Systems (ACM SIGOPS). ACM SOCC will be held in conjunction with ACM SIGMOD and ACM SOSP Conferences in alternate years, starting with ACM SIGMOD in 2010.

The scope of SOCC Symposia will be broad and will encompass diverse systems topics such as software as a service, virtualization, and scalable cloud data services. Many facets of systems and data management issues will need to be revisited in the context of cloud computing. Topics of interest include but are not limited to:

- Administration and Manageability
- Data Privacy
- Data Services Architectures
- Distributed and Parallel Query Processing
- Energy Management
- Geographic Distribution
- Grid Computing
- High Availability and Reliability
- Infrastructure Technologies
- Large Scale Cloud Applications
- Multi-tenancy
- Provisioning and Metering
- Resource management and Performance
- Scientific Data Management
- Security of Services
- Service Level Agreements
- Storage Architectures
- Transactional Models
- Virtualization Technologies

The final program, registration, and location information can be found at: http://research.microsoft.com/socc2010.

In addition to a technical program of research papers, position papers and industrial presentations, the program features three keynote talks from leading architects of Cloud systems:

**Evolution and Future Directions of Large-Scale Storage and Computation Systems at Google, Jeffrey Dean (Google)**

**Building Facebook: Performance at Massive Scale, Jason Sobel (Facebook)**

**The Internal Design of Salesforce.com's Multi-Tenant Architecture, Rob Woollen (Salesforce.com)**

ACM SOCC conference will be held at the Hyatt Regency Hotel, Indianapolis, IN. SOCC is co-located with ACM SIGMOD/PODS, and shares a registration form, allowing you to register for either one or both in one convenient place.

## *Important Dates*

| | |
|---|---|
| **Early Registration:** | May 7, 2010 (11:59pm, PST) |
| **Hotel discount cut-off:** | May 15, 2010 |

## *Conference Officers*

**General Chair:**
Joseph M. Hellerstein, U. C. Berkeley
**Program Chairs:**
Surajit Chaudhuri, Microsoft Research
Mendel Rosenblum, Stanford University
**Steering Committee:**
Phil Bernstein, Microsoft Research
Ken Birman, Cornell University
Joseph M. Hellerstein, U. C. Berkeley
John Ousterhout, Stanford University
Raghu Ramakrishnan, Yahoo! Research
Doug Terry, Microsoft Research
John Wilkes, Google
**Publicity Chair:**
Aman Kansal, Microsoft Research
**Treasurer:**
Brian Cooper, Yahoo! Research

## *Program Committee*

Anastasia Ailamaki, EPFL
Brian Bershad, Google
Ken Birman, Cornell University
Michael Carey, UC Irvine
Felipe Cabrera, Amazon
Jeff Chase, Duke
Byung-gon Chun, Intel Labs Berkeley
Dilma M da Silva, IBM
David Dewitt, Microsoft
Shel Finkelstein, SAP
Armando Fox, UC Berkeley
Tal Garfinkel, Stanford
Alon Halevy, Google
James Hamilton, Amazon
Jeff Hammerbacher, Cloudera
Joe Hellerstein, UC Berkeley
Alfons Kemper, TU München
Donald Kossman, ETH
Orran Krieger, VMware
Jim Larus, Microsoft
Jeffrey Naughton, UWisconsin, Madison
Christopher Olston, Yahoo
Hamid Pirahesh, IBM
Raghu Ramakrishnan, Yahoo!
Venugopalan Ramasubramanian, MSR
Hovav Shacham, UCSD
Donovan Schneider, Salesforce.com
Andy Warfield, U. British Columbia
Hakim Weatherspoon, Cornell
John Wilkes, Google

## Looking for Computer Science
## Undergraduate and Graduate Students

**DataBase Mentoring (DB Me) Workshop at SIGMOD 2010**
Location: Indianapolis, IN, USA
Date: Friday, June 11, 2010
***http://www.cs.ubc.ca/~rap/dbme/***

We proudly announce the SIGMOD CRA-W/CDC DB Me(ntoring) Workshop. The main goal of this workshop is to encourage current undergraduate and graduate students (and in particular *women* and *underrepresented minorities*) to obtain a PhD in computer science or computer engineering. The specific theme of this workshop is Database Systems.

This workshop will include a combination of technical and mentoring sessions led by internationally known researchers. Technical sessions will include hot topics and future directions in several areas of databases and their significance to solving real world problems. Mentoring sessions will focus on motivating and preparing students to go to graduate school, excel in graduate school, and make a contribution to the research and development of database technology. All sessions will include a question and answer period.

***There is no charge for this workshop. Students may apply to be reimbursed for all reasonable travel expenses.*** Any student may apply to the workshop, but first priority will be given to women and underrepresented minority applicants.

**APPLICATIONS ARE DUE ON MARCH 31**. Apply online at: http://www.cs.ubc.ca/~rap/dbme/ Questions can be e-mailed to db-me@cs.ubc.ca.

# Call for Participation

## DBTEST 2010: Third International Workshop on Testing Database Systems

Monday, 7 June 2010, Indianapolis, Indiana
In conjunction with SIGMOD/PODS 2010

`http://www.cs.duke.edu/dbtest2010`

New usage patterns, evolving hardware trends, and increased competition drive continuous innovation and expansion of data-processing systems. Commercial database vendors are adding new features related to ease of management, semistructured data, data compression, parallelism, reliability, and security. New data-processing systems are being developed over MapReduce backends, key-value stores, low-power devices, and widely-distributed systems. It is increasingly expensive to test and tune these systems. DBTest 2010 aims to bring together researchers and practitioners to discuss important open problems and new techniques in testing systems for data management.

### Topics of Interest

Testing techniques for data-processing systems, storage services, and applications

Data and workload generation for testing

Generation of stochastic models for large test matrices

Algorithms for automatic program verification

Techniques to maximize code coverage during testing

Testing the reliability and availability of data-processing systems

Testing and designing systems that are robust to estimation inaccuracies

Testing the effectiveness of adaptive policies and components

Interactions between testing and tuning of data-processing systems

Testing data consistency Vs. availability tradeoffs

Metrics for predictability of query and workload performance

Metrics for query plan robustness

Security and vulnerability testing

War stories and vision papers

The Workshop program will be designed to foster discussion amongst the participants. The DBTEST 2010 program will offer both a panel discussion that will include panelists from both industry and academia, along with a keynote address by Phyllis Frankl of the Computer and Information Science Department, Polytechnic University, New York.

You can register for the Workshop and obtain hotel and conference information through the ACM SIGMOD website at `http://www.sigmod2010.org.`

WORKSHOP CO-CHAIRS:

SHIVNATH BABU, *Duke University (USA)*

GLENN PAULLEY, *Sybase iAnywhere (Canada)*

PROGRAM COMMITTEE:

SHIVNATH BABU, *Duke University (USA)*

CARSTEN BINNIG, *SAP (Germany)*

JOSE BLAKELEY, *Microsoft (USA)*

MITCH CHERNIACK, *Brandeis University (USA)*

LEONIDAS GALANIS, *Oracle (USA)*

DAVE GODWIN, *IBM Toronto Laboratory (Canada)*

HARYADI GUNAWI, *University of California, Berkeley (USA)*

DONALD KOSSMANN, *ETH Zurich (Switzerland)*

VIVEK NARASAYYA, *Microsoft Research (USA)*

GLENN PAULLEY, *Sybase iAnywhere (Canada)*

KEN SALEM, *University of Waterloo (Canada)*

JIRI SCHINDLER, *Network Appliance (USA)*

STEERING COMMITTEE:

LEO GIAKOUMAKIS, *Microsoft (USA)*

DONALD KOSSMANN, *ETH Zurich (Switzerland)*

INDUSTRY SPONSORS:

GREENPLUM

IBM

ORACLE

MICROSOFT

SYBASE

IDAR
SIGMOD 2010

# PhD Workshop on Innovative Database Research

Indianapolis, Indiana, USA
June 11th, 2010

## WORKSHOP ORGANIZERS

### CO-CHAIRS

Mirella M. Moro, Univ. Fed. Minas Gerais, Brazil
Zografoula Vagena, Univ. Southern Denmark, Denmark

### STEERING COMMITTEE MEMBERS

Tok W. Ling, Nat. Univ. of Singapore, Singapore
Xiaofeng Meng, Renmin Univ. of China, China
Ge Yu, Northeastern University, China

### PROGRAM COMMITTEE

Karl Aberer, EPFL, Switzerland
Marcelo Arenas, Pontificia Univ. Catolica de Chile, Chile
Vanessa Braganholo, Univ. Fed. Rio de Janeiro, Brazil
Carlotta Domeniconi, George Mason University, USA
Carina Dorneles, Univ. Fed. Santa Catarina, Brazil
Renata Galante, Univ. Fed. Rio Grande do Sul, Brazil
Mingsheng Hong, Vertica, USA
Wynne Hsu, National Univ. of Singapore, Singapore
Verena Kantere, EPFL, Switzerland
Anastasios Kementsietsidis, IBM T.J. Watson R. C., USA
Konstantinos Krikellas, Greenplum, USA
Harumi A. Kuno, Hewlett-Packard Laboratories, USA
Lipyeow Lim, University of Hawaii at Manoa, USA
Song Lin, Yahoo!, USA
Jiaheng Lu, Renmin University of China, China
Gerome Miklau, University of Massachusetts, USA
Mario A. Nascimento, University of Alberta, Canada
Thomas Neumann, Max Planck Institut Inf., Germany
Fatma Özcan, IBM Almaden Research, USA
Josep Pujol, Telefonica Research, Spain
Rodrigo Schmidt, Facebook, USA
Stefan Schönauer, University of Helsinki, Finland
Heng Tao Shen, University of Queensland, Australia
Divesh Srivastava, AT&T, USA
Garret Swart, Oracle, USA
Stratis Viglas, University of Edinburgh, UK
Michail Vlachos, IBM Zürich Research Lab., Switzerland
Milan Vojnovic, Microsoft Research, UK
Haixun Wang, Microsoft Research Asia, China
Fan Yang, Google, USA
Dimitris Zeinalipour-Yazti, University of Cyprus, Cyprus
Chun Zhang, ArcSight Inc, USA
Yongluan Zhou, Univ. Southern Denmark, Denmark

ACM SIGMOD INDIANAPOLIS, USA, 2010

The Ph.D. Workshop on Innovative Data base Research (to be held in cooperation with SIGMOD) is intended to bring together Ph.D. students working on t opics related to the SIGMOD conference series. The workshop offers Ph.D. students the opportunity to present, discuss, and receive feedback on their research in a constructive and international atmosphere. The workshop will be accompanied by prominent professors, researchers and practitioners in the fields of data and information management, who will participate actively and contribute to the discussions. The workshop is co-located with SIGMOD 2010 and will take place on Friday June 11th, 2009.

## ACCEPTED PAPERS

- Building a Power-Aware Database Management System. *Zichen Xu, University of South Florida, USA*
- Event Sequence Processing: New Models and Optimization Techniques. *Mo Liu and Elke Rundensteiner, Worcester Polytechnic Institute, USA*
- Exploiting Locality for Query Processing and Compression in Scientific Databases. *Anand Kumar, University of South Florida, USA*
- Improved Approaches To Mine Rare Association Rules in Transactional Databases. *R. Uday Kiran and P. Krishna Reddy, International Institute of Information Technology-Hyderabad, India*
- Multiple Relationship Based Deduplication. *Pei Li, University of Milan, Bicocca, Italy*
- Put All Eggs in One Basket: an OLTP and OLAP Database Approach for Traceability Data. *Veneta Dobreva and Martina-Cezara Albutiu, Technische Universität München, Germany*
- Specification and Verification of Web Services Transactions. *Iman Saleh, Virginia Polytechnic Institute and State University, USA*
- Statistical Modeling of Large Distribution Sets. *Yasuko Matsubara, Yasushi Sakurai and Masatoshi Yoshikawa, Kyoto University, Japan*
- Unsupervised Strategies for Information Extraction by Text Segmentation. *Eli Cortez and Altigran S. da Silva, Universidade Federal do Amazonas, Brazil*

## KEYNOTE SPEAKER

**Divesh Srivastava** is the head of Database Research at AT&T Labs Research. He received his Ph.D. from the University of Wisconsin, Madison, and his Bachelor of Technology from the Indian Institute of Technology, Bombay, India. His current research interests include data quality, data streams and data privacy.

## REGISTRATION

There is no registration fee for IDAR 2010. The space is limited. If you pre-register for these workshops and do not attend, you will be charged $80 to cover the cost of food and materials.

Early bird registration for SIGMOD is due May 15th. You can register through the webpage at https://regmaster3.com/2010conf/MOD10/register.php

http://www.dcc.ufmg.br/idar2010

WANDS'10

ACM SIGMOD

# CALL FOR PARTICIPATION

## 1st International Workshop on Workflow Approaches to New Data-centric Science

**Held in conjunction with SIGMOD 2010, 10th June 2010, Indianapolis, IN, USA**

**Registration website:**

**https://regmaster3.com/2010conf/MOD10/register.php**

A number of innovative, but uncoordinated, efforts in data-centric workflows have made their mark on the scientific and business world in recent years. The goal of this workshop is to use these efforts to bring together a research community around the theoretical foundations, technology development, and domain applications of workflow systems. As a first postulate, we take that there is no uniform workflow solution, in the same way that there is no uniform programming language. Furthermore, our goal is not to promote a particular technology, or even interoperability between individual technologies. What we intend to achieve is to present, discuss and ultimately better the different computational models, usability patterns, and domain approaches used in the field.

The relevant topics include, and are not limited to, the following:

- Role of workfows in data analytics, mining and statistics
- Performance estimation and optimization of workflow execution
- Scalability of workflow-based solutions on very large data sets
- Role of metadata in static profiling of dataflows
- Impact of, and new requirements for, workflow technology on open science
- Workflow warehousing, indexing, searching, and mining
- Workflows for service orchestration
- Mapping workflows to cloud computing environments
- In-database workflow execution
- User interaction models for complex eScience: visibility vs. commoditization
- Workflow repurposing and reuse
- Privacy models for data-centric workflows
- Workflow semantics
- Workflows as invisible middleware
- Workflows in Service-oriented life sciences

This year's program includes six research talks, a project report, and two keynote speeches: by Prof. Bertam Ludaescher from UC Davis and Prof. Tova Milo, from Tel Aviv University. We are looking forward to seeing you there and having an exciting day!

**Partly supported by:**

DataONE

**PRELIMINARY PROGRAM**

**8.30am – 10.00am Opening and First Keynote**

- Keynote Speaker: Bertram Ludaescher, University of California, Davis

*10.00am – 10.30am Coffee break*

**10.30am – 12.00pm Paper session**

- Exploring Repositories of Scientific Workflows, Julia Stoyanovich, Ben Taskar and Susan Davidson
- Privacy Issues in Scientific Workflow Provenance, Susan Davidson, Sanjeev Khanna, Sudeepa Roy and Sarah Cohen Boulakia.
- DFL designer - collection-oriented scientific workflows with Petri nets and nested relational calculus, Jacek Sroka, Piotr Wlodarczyk, Lukasz Krupa and Jan Hidders.

*12.00pm – 1.30pm Lunch*

**1.30pm – 3.00pm Paper session**

- A Multi-Dimensional Classification Model for Scientific Workflow Characteristics, Lavanya Ramakrishnan and Beth Plale.
- A General Approach to Data-Intensive Computing using the Meandre Component-Based Framework, Bernie Ács, Xavier Llorà, Loretta Auvil, Boris Capitanu, David Tcheng, Mike Haberman, Limin Dong, Tim Wentling and Michael Welge.
- Meta-Workflows: Pattern-based Interoperability between Galaxy and Taverna, Mohamed Abouelhoda, Shady Alaa and Moustafa Ghanem.

*3.00pm – 3.30pm Coffee break*

**3.30pm – 5.00pm Second Keynote, Discussion**

- Report from DCW2009, NSF Workshop on Data-Centric Workflows. Rick Hull, IBM.
- Keynote Speaker: Tova Milo, University of Tel Aviv

Program Chairs:

Paolo Missier, pmissier@cs.man.ac.uk
Vasa Curcin, vasa.curcin@imperial.ac.uk
Susan Davidson, susan@seas.upenn.edu

# WebDB 2010 at ACM SIGMOD 2010
# 13th International Workshop on the Web and Databases

June 6, 2010. Indianapolis, Indiana `webdb2010.org`

Xin Luna Dong
AT&T Labs-Research, USA
lunadong@research.att.com

Felix Naumann
Hasso-Plattner-Institute (HPI), Germany
naumann@hpi.uni-potsdam.de

The WebDB workshop focuses on providing a forum where researchers, theoreticians, and practitioners can share their knowledge and opinions about problems and solutions at the intersection of data management and the Web. WebDB has high impact and has been a forum in which a number of seminal papers have been presented.
The workshop chairs welcome researchers and practitioners to register for the workshop at `http://webdb2010.org/registration.html`. The registration fee is $100; rebates for students ($65) and ACM members ($80) are available. Breakfast, lunch, and coffee breaks will be provided.

## 1. TOPICS OF INTEREST

This year's WebDB will focus on *Quality of Web Data* and on *Linked Data*, but papers on all aspects of the Web and Databases were welcome.

- Bridging structured and unstructured data on the Web
- Cleansing and Integrating Linked Data and Web data
- Cloud and distributed computing over the Web
- Data integration over the Web
- Data models and query languages for Web information
- Data-intensive applications on the Web
- Database support for social network and Web 2.0
- Information retrieval in semi-structured data
- Location-aware Web applications
- Mining Web data
- Profiling Linked Data
- Producing and Querying Linked Data
- Quality of Web data and Linked Data
- Semantic search on the Web
- Semi-structured data management
- The Semantic Web and reasoning on Web data
- Web information extraction
- Web privacy and security
- Web services-based architecture and applications
- Web source discovery, analysis, and retrieval
- Web-based distributed data management

## 2. PROGRAM

A special highlight of this year's WebDB is the invited speaker, *Chris Bizer* from Free University of Berlin. Chris is cocounder of the DBpedia project which publishes structured and cleansed data extracted from many language versions of Wikipedia. DBpedia has been the kernel for the much larger ambition of creating a enormous web of Linked Data.

At the deadline for this issue of SIGMOD Record, the review process was still ongoing. The program will feature the best papers from a very large amount of submitted papers.

## 3. PROGRAM COMMITTEE

- Co-chair: Xin Luna Dong, AT&T Labs-Research, USA
- Co-chair: Felix Naumann, Hasso Plattner Institute (HPI), Germany
- Alexander Löser (University of Technology, Berlin)
- Alin Deutsch (UC San Diego)
- Amelie Marian (Rutgers University)
- AnHai Doan (University of Wisconsin-Madison)
- Anish Das Sarma (Yahoo!)
- Beng Chin Ooi (National University of Singapore)
- Bertram Ludäscher (UC Davis)
- Bin Cui (Beijing University)
- Chris Bizer (Free University of Berlin)
- Christoph Koch (Cornell University)
- Christopher Ré (University of Wisconsin-Madison)
- Dan Suciu (University of Washington)
- Daniela Florescu (Oracle)
- Donald Kossmann (ETH-Zürich)
- Erhard Rahm (University of Leipzig)
- Gerome Miklau (UMass-Amherst)
- Haixun Wang (Microsoft Research at Asia)
- Jayavel Shanmugasundaram (Yahoo!)
- Jeffrey Yu (Chinese University of Hong Kong)
- Karl Aberer (EPFL)
- Michael Benedikt (University of Oxford)
- Michael Cafarella (University of Michigan)
- Michalis Petropoulos (SUNY Buffalo)
- Panagiotis Ipeirotis (NYU)
- Piero Fraternalli (Politecnico di Milano)
- Todd Green (UC Davis)
- Vassilis Christophisdes (ICS-FORTH)
- Wenfei Fan (University of Edinburgh)
- Yi Chen (Arizona State University)