SIGMOD Officers, Committees, and Awardees

Chair

Donald Kossmann
Systems Group
ETH Zürich
Cab F 73
8092 Zuerich
SWITZERLAND
+41 44 632 29 40
<donaldk AT inf.ethz.ch>

Vice-Chair

Anastasia Ailamaki
School of Computer and
Communication Sciences, EPFL
EPFL/IC/IIF/DIAS
Station 14, CH-1015 Lausanne
SWITZERLAND
+41 21 693 75 64
<natassa AT epfl.ch>

Secretary/Treasurer

Magdalena Balazinska
Computer Science & Engineering
University of Washington
Box 352350
Seattle, WA
USA
+1 206-616-1069
<magda AT cs.washington.edu>

SIGMOD Executive Committee:
Donald Kossmann (Chair), Anastasia Ailamaki (Vice-Chair), Magdalena Balazinska, K. Selçuk Candan, Yanlei Diao, Curtis Dyreson, Christian Jensen, Yannis Ioannidis, and Tova Milo.

Advisory Board:
Raghu Ramakrishnan (Chair, Microsoft), Amr El Abbadi, Serge Abiteboul, Ricardo Baeza-Yates, Phil Bernstein, Elisa Bertino, Mike Carey, Surajit Chaudhuri, Christos Faloutsos, Alon Halevy, Joe Hellerstein, Renée Miller, C. Mohan, Beng-Chin Ooi, Z. Meral Ozsoyoglu, Sunita Sarawagi, Min Wang, and Gerhard Weikum.

SIGMOD Information Director:
Curtis Dyreson, Utah State University <curtis.dyreson AT usu.edu>

Associate Information Directors:
Manfred Jeusfeld, Georgia Koutrika, Wim Martens, Mirella Moro, Rachel Pottinger, and Jun Yang.

SIGMOD Record Editor-in-Chief:
Yanlei Diao, University of Massachusetts Amherst <yanlei AT cs.umass.edu>

SIGMOD Record Associate Editors:
Pablo Barceló, Vanessa Braganholo, Marco Brambilla, Chee Yong Chan, Rada Chirkova, Anish Das Sarma, Alkis Simitsis, Nesime Tatbul, and Marianne Winslett.

SIGMOD Conference Coordinator:
K. Selçuk Candan, Arizona State University

PODS Executive Committee: Rick Hull (Chair, IBM Research), Michael Benedikt, Wenfei Fan, Martin Grohe, Maurizio Lenzerini, Jan Paradaens.

Sister Society Liaisons:
Raghu Ramakhrishnan (SIGKDD), Yannis Ioannidis (EDBT Endowment), Christian Jensen (IEEE TKDE).

Awards Committee:
Umesh Dayal (Chair, Hitachi America Ltd.), Elisa Bertino, Surajit Chaudhuri, Masaru Kitsuregawa, and Maurizio Lenzerini.

Jim Gray Doctoral Dissertation Award Committee:
Tova Milo (Co-Chair, Tel Aviv University), Timos Sellis (Co-Chair, RMIT University), Ashraf Aboulnaga, Sudipto Das, Juliana Freire, Minos Garofalakis, Dan Suciu, Kian-Lee Tan.

[Last updated : September 30th, 2014]

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. Formerly known as the "SIGMOD Innovations Award", it now honors 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)
Umeshwar Dayal (2010)
Surajit Chaudhuri (2011)
Bruce Lindsay (2012)
Stefano Ceri (2013)
Martin Kersten (2014)

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)
David Lomet (2010)
Gerhard Weikum (2011)
Marianne Winslett (2012)
H.V. Jagadish (2013)
Kyu-Young Whang (2014)

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.* Recipients of the award are the following:

- **2006** *Winner*: Gerome Miklau, University of Washington. *Runners-up*: Marcelo Arenas and Yanlei Diao.
- **2007** *Winner*: Boon Thau Loo, University of California at Berkeley. *Honorable Mentions*: Xifeng Yan and Martin Theobald.
- **2008** *Winner*: Ariel Fuxman, University of Toronto. *Honorable Mentions*: Cong Yu and Nilesh Dalvi.
- **2009** *Winner*: Daniel Abadi, MIT. *Honorable Mentions*: Bee-Chung Chen and Ashwin Machanavajjhala.
- **2010** *Winner:* Christopher Ré, University of Washington. *Honorable Mentions*: Soumyadeb Mitra and Fabian Suchanek.
- **2011** *Winner*: Stratos Idreos, Centrum Wiskunde & Informatica. *Honorable Mentions*: Todd Green and Karl Schnaitterz.
- **2012** *Winner*: Ryan Johnson, Carnegie Mellon University. *Honorable Mention*: Bogdan Alexe.
- **2013** *Winner*: Sudipto Das, University of California, Santa Barbara. *Honorable Mention*: Herodotos Herodotou and Wenchao Zhou.
- **2014** *Winners*: Aditya Parameswaran, Stanford University, and Andy Pavlo, Brown University.

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

[Last updated : September 30th, 2014]

Editor's Notes

Welcome to the September 2014 issue of the ACM SIGMOD Record!

The issue opens with a Database Principles article by Deutsch, Hull, and Vianu on automatic verification of database-centric systems. This work is motivated by the observation that the emerging high-level specification tools for database-centric systems provide a natural target for verification. The results described by this article suggest that verification may indeed be feasible for significant classes of database-driven systems, specifically under the two representative models of business artifacts and database-driven web services where restrictions can be placed to guarantee decidability of verification. The article concludes by outlining several directions for future work, including both theoretical and practical aspects. This article introduces the reader to an important research topic, and also facilitates future research at the confluence of database and computer-aided verification areas.

The Research and Vision Articles Column features an article by Guo, Jensen, and Yang on data management issues in the transportation sector. The article is driven by the vision that continued proliferation of sensors and mobile devices, combined with the drive towards open data, will result in rapidly increasing volumes of transportation data. Furthermore, efficient and effective analysis of "big transportation data" will enable people to extract new, important transportation knowledge. Towards this goal, this article describes the transportation data, presents key challenges related to the extraction of thorough, timely, and trustworthy traffic knowledge to achieve total traffic awareness, and closes by outlining the services that will be enabled. Overall, this article provides a good introduction to big transportation data, with the aim to motivate more researchers to work on the related data management problems.

The Surveys Column features two articles. Braganholo and Mattoso provide a survey on XML fragmentation. This is a key issue in distributed (e.g., MapReduce) processing where large volumes of XML data are split into pieces and distributed to different compute nodes, and queries are split accordingly to exploit parallel processing. However, there is no consensus in the database literature on what an XML fragment is. This article surveys different XML fragmentation techniques, discussing their features and highlighting their limitations. Such discussion will enable the XML user to choose the appropriate technique based on the target query-processing scenario. The second article, by Santos, Bernardino, and Vieira, surveys Database Intrusion Detection Systems (DIDS). It presents a variety of existing intrusion detection techniques, and discusses how these IDIS are applied in each database context, pointing out their weaknesses given typical user workloads and characteristics of the environment. This discussion is followed by the identification of a set of challenges and opportunities, as well as requirements and guidelines to drive the development of new or improved DIDS.

The Systems and Prototypes Column features an integrated system for mining, querying, and managing Web document corpora, developed by Mousavi, Atzori, Gao, and Zaniolo. This system is designed for Wikipedia's InfoBoxes, which provide the main knowledge source for many applications on the Web while suffering from incompleteness, inconsistencies, and inaccuracies. To overcome these problems, the proposed system integrates the *IBminer* system that extracts InfoBox information by text-mining Wikipedia pages; the *IKBStore* system that integrates mined text with other sources to build a knowledge base; and a user-friendly interface that supports knowledge editing and query-by-example functions needed for managing a knowledge base.

In the Research Centers Column, Kemper and Neumann outline the research agenda of the database group at Technische Universität München (TUM). The research at TUM has a long-tem goal of developing a high-performance database engine that unites OLTP and OLAP into a single system. For this purpose, a hybrid main memory database system, called HyPer, has been developed. HyPer supports both OLTP and

OLAP applications on the same database state by exploiting the OS/processor-support for virtual memory management. This article describes the key design choices and reviews recent work on a variety of topics, including NUMA-aware many-core parallelism, in-memory database compression, indexing on main-memory databases, and lock-free synchronization by hardware transactional memory. This line of research has led to a number of awards granted to the members of the TUM database team.

This issue features two event reports. First, it is our pleasure to include the Beckman Report on Database Research. Every few years a group of database researchers meet to discuss the state of database research, its impact on practice, and important new directions. This report summarizes the discussion and conclusions of the eighth such meeting, held October 14-15, 2013 in Irvine California. The meeting identified Big Data as a defining challenge of our time, and argued that enormous opportunities are presented to the database community to make transformative impact. In particular, this report discusses research challenges in five areas: scalable big/fast data infrastructures; coping with diversity in the data management landscape; end-to-end processing and understanding of data; cloud services; and managing the diverse roles of people in the data life cycle. The article also discusses community challenges, in particular, the need to rethink our approach to teaching data management technologies. The second article in the column, by Varde and Tatti, reports on the PhD Forum held at the IEEE International Conference on Data Mining (ICDM) 2013. The report surveys the papers from the forum, covering both core research and applied research in data mining, and discusses open issues that provide a scope for further research.

This issue closes with the call for papers for EuroSys 2015, to be held April 21-24, 2015 in Bordeaux, France, and the call for participation for SoCC, to be held November 3-5, 2014 in Seattle, Washington.

On behalf of the SIGMOD Record Editorial board, I hope that you will all enjoy reading the September 2014 issue of the SIGMOD Record.

Your submissions to the Record are welcome via the submission site:
http://sigmod.hosting.acm.org/record

Prior to submitting, you are encouraged to read the Editorial Policy on the SIGMOD Record's Web site (http://www.sigmod.org/publications/sigmod-record/sigmod-record-editorial-policy).

Yanlei Diao

September 2014

Past SIGMOD Record Editors:

Harrison R. Morse (1969)
Daniel O'Connell (1971 – 1973)
Randall Rustin (1974-1975)
Douglas S. Kerr (1976-1978)
Thomas J. Cook (1981 – 1983)
Jon D. Clark (1984 – 1985)
Margaret H. Dunham (1986 – 1988)
Arie Segev (1989 – 1995)
Jennifer Widom (1995 – 1996)
Michael Franklin (1996 – 2000)
Ling Liu (2000 – 2004)
Mario Nascimento (2005 – 2007)
Alexandros Labrinidis (2007 – 2009)
Ioana Manolescu (2009-2013)

Automatic Verification of Database-Centric Systems*

Alin Deutsch
UC San Diego
deutsch@cs.ucsd.edu

Richard Hull
IBM T.J. Watson Research Center
hull@us.ibm.com

Victor Vianu
UC San Diego & INRIA Saclay
vianu@cs.ucsd.edu

1. INTRODUCTION

Software systems centered around a database are pervasive in numerous applications. They are encountered in areas as diverse as electronic commerce, e-government, scientific applications, enterprise information systems, and business process management. Such systems are often very complex and prone to costly bugs, whence the need for verification of critical properties.

Classical software verification techniques that can be applied to such systems include *model checking* and *theorem proving*. However, both have serious limitations. Indeed, model checking usually requires performing finite-state abstraction on the data, resulting in loss of semantics for both the system and properties being verified. Theorem proving is incomplete, requiring expert user feedback.

Recently, an alternative approach to verification of database-centric systems has taken shape, at the confluence of the database and computer-aided verification areas. It aims to identify restricted but sufficiently expressive classes of database-driven applications and properties for which sound and complete verification can be performed in a fully automatic way. This approach leverages another trend in database-driven applications: the emergence of high-level specification tools for database-centered systems, such as interactive web applications and data-driven business processes. We review next a few representative examples.

A commercially successful high-level specification tool for web applications is Web Ratio [1], an outgrowth of the earlier academic prototype WebML [20, 17]. Web Ratio allows to specify a Web application using an interactive variant of the E-R model augmented with a workflow formalism. Non-interactive variants of Web page specifications had already been proposed in Strudel [41], Araneus [60] and Weave [42], targeting the automatic generation of Web sites from an underlying database. High-level specification tools have also emerged in the area of business process management, concomitantly with an evolution from the traditional process-centric approach towards data awareness. A notable exponent of this class is the *business artifact model* pioneered in [65, 53], deployed by IBM in professional services. Business artifacts (or simply "artifacts") model key business-relevant entities, which are updated by a set of services that implement business process tasks. A collection of artifacts and services is called an *artifact system*. This modeling approach has been successfully deployed in practice [7, 6, 21, 27, 73], and has been adopted in the OMG standard for Case Management [9].

Tools such as the above automatically generate the database-centric application code from the high-level specification. This not only allows fast prototyping and improves programmer productivity but, as a side effect, provides new opportunities for automatic verification. Indeed, the high-level specification is a natural target for verification, as it addresses the most likely source of errors (the application's specification, as opposed to the less likely errors in the automatic generator's implementation).

The theoretical and practical results obtained so far concerning the verification of such systems are quite encouraging. They suggest that, unlike arbitrary software systems, significant classes of data-driven systems may be amenable to automatic verification. This relies on a novel marriage of database and model checking techniques, and is relevant to both the database and the computer-aided verification communities.

In this article, we describe several models and results on automatic verification of database-driven systems, focusing on temporal properties of their underlying workflows. To streamline the presentation, we focus on verification of business artifacts, and use it as a vehicle to introduce the main con-

*Database principles column. Column editor: Pablo Barceló. Dept. of Computer Science, University of Chile. E-mail: pbarcelo@dcc.uchile.cl.

cepts and results. Moreover, the technical challenges posed by verification of business artifacts are representative of those present in some of the other models (notably data-driven web services), which can be viewed as syntactic variants of business artifacts. We also summarize some of the work pertaining specifically to data-driven web services.

2. BUSINESS ARTIFACTS

IBM's business artifacts model key business-relevant entities, which are updated by a set of services that implement business process tasks. The notion of business artifact was first introduced in [65] and [53] (called there "adaptive documents"), and was further studied, from both practical and theoretical perspectives, in [6, 43, 44, 8, 56, 25, 47, 4]. (Some of these publications use the term "business entity" in place of "business artifact"). Some key roots of the artifact model are present in "adaptive business objects" [63], "business entities", "document-driven" workflow [72] and "document" engineering [45]. The Vortex framework [49, 38, 48] also allows the specification of database manipulations and provides declarative specifications for when services are applicable to a given artifact.

The artifact model is inspired in part by the field of semantic web services. In particular, the OWL-S proposal [59, 58] describes the semantics of services in terms of input parameters, output parameters, pre- and post-conditions. In the artifact model considered here the services are applied in a sequential fashion (there is no true concurrency). IBM has developed Siena [23], a tool for compiling artifact-based procedural specifications into code supporting the corresponding business process. Its open-source descendant, the BizArtifact suite [10], has just been announced. The Guard-Stage-Milestone (GSM) approach [25, 47] to artifact lifecycles permits services with pre- and post-conditions, parallelism, and hierarchy. The OMG standard for Case Management Model and Notation (CMMN) [9], announced last year, draws key foundational elements from GSM [57].

We next describe a minimalistic variant of the artifact model, adequate for illustrating the results on verification. The presentation is informal, relying mainly on a running example (the formal development is provided in [30, 24]). The example, modeling an e-commerce process, features several characteristics.

1. The system routinely queries an underlying database, for instance to look up the price of a product and the shipping weight restrictions.

2. The validity checks and updates carried out by the services involve arithmetic operations. For instance, to be valid, an order must satisfy such conditions as: (a) the product weight must be within the selected shipment method's limit, and (b) if the buyer uses a coupon, the sum of product price and shipping cost must exceed the coupon's minimum purchase limit.

3. Finally, the correctness of the business process relies on database integrity constraints. For instance, the system must check that a selected triple of product, shipment type and coupon are globally compatible. This check is implemented by several local tests, each running at a distinct instant of the interaction, as user selections become available. Each local test accesses distinct tables in the database, yet they globally refer to the same product, due to the keys and foreign keys satisfied by these tables.

The example models an e-commerce business process in which the customer chooses a product and a shipment method and applies various kinds of coupons to the order. There are two kinds of coupons: discount coupons subtract their value from the total (e.g. a $50 coupon) and free-shipment coupons subtract the shipping costs from the total. The order is filled in a sequential manner (first pick the product, then the shipment, then claim a coupon), as is customary on e-commerce web-sites. After the order is filled, the system awaits for the customer to submit a payment. If the payment matches the amount owed, the system proceeds to shipping the product.

As mentioned earlier, an artifact is an evolving record of values. The values are referred to by variables (sometimes called *attributes*). In general, an artifact system consists of several artifacts, evolving under the action of *services*, specified by pre- and post-conditions. This notion of service corresponds roughly to the notion of task in BPM (although it may also cover web services implementing tasks). For instance, in GSM terms, a service corresponds to a *stage*, a pre-condition to a *guard*, and a post-condition to a *milestone*.

In the example, we use a single artifact with the following variables:

`status`,`prod_id`,`ship_type`,`coupon`
`amount_owed` `amount_paid`,`amount_refunded`

The `status` variable tracks the status of the order and can take the following values:

"edit_product", "edit_ship", "edit_coupon"
"processing", "received_payment",
"shipping", "shipped", "canceling", "canceled".

Artifact variables `ship_type` and `coupon` record the customer's selection, received as an external input. `amount_paid` is also an external input (from the customer, possibly indirectly via a credit card service). Variable `amount_owed` is set by the system using arithmetic operations that sum up product price and shipment cost, subtracting the coupon value. Variable `amount_refunded` is set by the system in case a refund is activated.

The database includes the following tables, where underlined attributes denote keys. Recall that a key is a set of attributes that uniquely identify each tuple in a relation.

`PRODUCTS`(`id`, `price`, `availability`, `weight`)
`COUPONS`(`code`, `type`, `value`, `min_value`, `free_shiptype`)
`SHIPPING`(`type`, `cost`, `max_weight`)
`OFFERS`(`prod_id`, `discounted_price`, `active`)

The database also satisfies the following foreign keys:

`COUPONS`[`free_shiptype`] $\subseteq$ `SHIPPING`[`type`] and
`OFFERS`[`prod_id`] $\subseteq$ `PRODUCTS`[`id`].

The first inclusion dependency says that each `free_shiptype` value in the `COUPONS` relation is also a `type` value in the `SHIPPING` relation. The second dependency states that every `prod_id` value in the `OFFERS` is the actual `id` of a product in the `PRODUCTS` relation.

The starting configuration of every artifact system is constrained by an initialization condition, which here states that `status` is initialized to "edit_prod", and all other variables to "undefined". By convention, we model undefined variables using the reserved constant λ.

The services. Recall that artifacts evolve under the action of services. As in the Guard-Stage Milestone approach mentioned above, each service is specified declaratively by a pre-condition π and a post-condition ψ, here limited to existential first-order ($\exists$FO) sentences. The pre-condition refers to the current values of the artifact variables and the database. The post-condition ψ refers simultaneously to the current and *next* artifact values, as well as the database. In addition, both π and ψ may use arithmetic constraints on the variables, limited to linear inequalities over the rationals.

The services shown in Figure 1 model a few of the business process tasks of the example. Throughout the example, we use primed artifact variables x' to refer to the *next* value of variable x.

Notice that the pre-conditions of the services check the value of the `status` variable. For instance, according to **choose_product**, the customer can only input her product choice while the order is in "edit_prod" status.

Also notice that the post-conditions constrain the next values of the artifact variables (denoted by a prime). For instance, according to **choose_product**, once a product has been picked, the next value of the status variable is "edit_shiptype", which will at a subsequent step enable the **choose_shiptype** service (by satisfying its pre-condition). Similarly, once the shipment type is chosen (as modeled by service **choose_shiptype**), the new status is "edit_coupon", which enables the **apply_coupon** service. The interplay of pre- and post-conditions achieves a sequential filling of the order, starting from the choice of product and ending with the claim of a coupon.

A post-condition may refer to both the current and next values of the artifact variables. For instance, in service **choose_shiptype**, the fact that only the shipment type is picked while the product remains unchanged, is modeled by preserving the product id: the next and current values of the corresponding artifact variable are set equal.

Pre- and post-conditions may query the database. For instance, in service **choose_product**, the post-condition ensures that the product id chosen by the customer is that of an available product (by checking that it appears in a `PRODUCTS` tuple, whose availability attribute is positive).

Finally, notice the arithmetic computation in the post-conditions. For instance, in service **apply_coupon**, the sum of the product price p and shipment cost c (looked up in the database) is adjusted with the coupon value (notice the distinct treatment of the two coupon types) and stored in the `amount_owed` artifact variable.

Observe that the first post-condition disjunct models the case when the customer inputs no coupon number (the next value `coupon`$'$ is set to undefined), in which case a different owed amount is computed, namely the sum of price and shipping cost.

Semantics The semantics of an artifact system $\mathcal{A}$ consists of its *runs*. Given a database D, a run of $\mathcal{A}$ is an infinite sequence $\{\rho_i\}_{\geq 0}$ of artifact records such that ρ_0 and D satisfy the initial condition of the system, and for each $i \geq 0$ there is a service S of the system such that ρ_i and D satisfy the pre-condition of S and ρ_i, ρ_{i+1} and D satisfy its post-condition. For uniformity, blocking prefixes of runs are extended to infinite runs by repeating forever their last record.

choose_product: The customer chooses a product.
π : status = "edit_prod"
$\psi : \exists p, a, w(\texttt{PRODUCTS}(\texttt{prod\_id}', p, a, w) \wedge a > 0) \wedge \texttt{status}' =$ "edit_shiptype"
choose_shiptype: The customer chooses a shipping option.
π : status = "edit_ship"
$\psi : \exists c, l, p, a, w(\texttt{SHIPPING}(\texttt{ship\_type}', c, l) \wedge \texttt{PRODUCTS}(\texttt{prod\_id}, p, a, w) \wedge l > w) \wedge$
$\texttt{status}' =$ "edit_coupon" $\wedge\, \texttt{prod\_id}' = \texttt{prod\_id}$
apply_coupon: The customer optionally inputs a coupon number.
π : status = "edit_coupon"
$\psi : (\texttt{coupon}' = \lambda \wedge \exists p, a, w, c, l(\texttt{PRODUCTS}(\texttt{prod\_id}, p, a, w) \wedge$
$\texttt{SHIPPING}(\texttt{ship\_type}, c, l) \wedge \texttt{amount\_owed}' = p + c) \wedge \texttt{status}' =$ "processing"
$\wedge \texttt{prod\_id}' = \texttt{prod\_id} \wedge \texttt{ship\_type}' = \texttt{ship\_type}) \vee$
$(\exists t, v, m, s, p, a, w, c, l(\texttt{COUPONS}(\texttt{coupon}', t, v, m, s) \wedge$
$\texttt{PRODUCTS}(\texttt{prod\_id}, p, a, w) \wedge \texttt{SHIPPING}(\texttt{ship\_type}, c, l) \wedge p + c \geq m \wedge$
$(t =$ "free_shipping" $\rightarrow (s = \texttt{ship\_type} \wedge \texttt{amount\_owed}' = p)) \wedge$
$(t =$ "discount" $\rightarrow \texttt{amount\_owed}' = p + c - v))$
$\wedge \texttt{status}' =$ "processing" $\wedge\, \texttt{prod\_id}' = \texttt{prod\_id} \wedge \texttt{ship\_type}' = \texttt{ship\_type})$

Figure 1: Three services

Note that the above semantics only considers linear runs of the system. A more informative notion is the *tree of runs* that completely captures the choice of services applicable at any given stage in the computation. We confine ourselves to linear runs because we are interested in verifying linear-time properties of the system. Formulating and verifying branching-time properties would require a semantics consisting of the full tree of runs.

The business process in the example exhibits a flexibility that, while desirable in practice for a postive customer experience, yields intricate runs, all of which need to be considered in verification. For instance, at any time before submitting a valid payment, the customer may edit the order (select a different product, shipping method, or change/add a coupon) an unbounded number of times. Likewise, the customer may cancel an order for a refund even after submitting a valid payment.

3. SPECIFYING TEMPORAL PROPERTIES OF DATA-CENTRIC SYSTEMS

We are interested in verifying temporal properties of runs of data-centric systems such as business artifacts. For instance, in our artifact system example, we would like to express such desiderata as:

> If a correct payment is submitted then at some time in the future either the product is shipped or the customer is refunded the correct amount.
>
> A free shipment coupon is accepted only if the available quantity of the product is greater than zero, the weight of the product is in the limit allowed by the shipment method, and the sum of price and shipping cost exceeds the coupon's minimum purchase value.

Similar properties are of interest for the data-driven web services described in Section 5. In order to specify such temporal properties we use an extension of LTL (linear-time temporal logic). Recall that LTL is propositional logic augmented with temporal operators such as **G** (always), **F** (eventually), **X** (next) and **U** (until) (e.g., see [66]). For example, $\mathbf{G}p$ says that p holds at all times in the run, $\mathbf{F}p$ says that p will eventually hold, and $\mathbf{G}(p \rightarrow \mathbf{F}q)$ says that whenever p holds, q must hold sometime in the future. The extension of LTL that we use, called[1] LTL-FO, is obtained from LTL by replacing propositions with quantifier-free FO statements about particular artifact records in the run. The statements use the artifact variables and may use additional *global* variables, shared by different statements and allowing to refer to values in different records. The global variables are universally quantified over the entire property.

For example, suppose we wish to specify the property that if a correct payment is submitted then at some time in the future either the product is shipped or the customer is refunded the correct amount. The property is of the form $\mathbf{G}(p \rightarrow \mathbf{F}q)$,

[1]The variant of LTL-FO used here differs from previous ones in that the FO formulas interpreting propositions are quantifier-free. By slight abuse we use here the same name.

where p says that a correct payment is submitted and q states that either the product is shipped or the customer is refunded the correct amount. Moreover, if the customer is refunded, the amount of the correct payment (given in p) should be the same as the amount of the refund (given in q). This requires using a global variable x in both p and q. More precisely, p is interpreted as the formula $\texttt{amount\_paid} = x \wedge \texttt{amount\_paid} = \texttt{amount\_owed}$ and q as $\texttt{status} = \text{"shipped"} \vee \texttt{amount\_refunded} = x$. This yields the LTL-FO property

$$(\varphi_1)\ \ \forall x \mathbf{G}((\texttt{amount\_paid} = x \wedge \texttt{amount\_paid} = \texttt{amount\_owed}) \rightarrow \mathbf{F}(\texttt{status} = \text{"shipped"} \vee \texttt{amount\_refunded} = x))$$

Note that, as one would expect, the global variable x is universally quantified at the end. We say that an artifact system $\mathcal{A}$ satisfies an LTL-FO sentence φ if all runs of the artifact system satisfy φ for all values of the global variables. Note that the database is fixed for each run, but may be different for different runs.

We now show a second property φ_2 for the running example, expressed by the LTL-FO formula

$$(\varphi_2)\ \ \forall v, m, s, p, a, w, c, l\ (\mathbf{G}((\texttt{prod\_id} \neq \lambda \wedge \texttt{ship\_type} \neq \lambda \wedge \texttt{COUPONS}(\texttt{coupon}, \text{"free\_ship"}, v, m, s) \wedge \texttt{PRODUCTS}(\texttt{prod\_id}, p, a, w) \wedge \texttt{SHIPPING}(\texttt{ship\_type}, c, l)) \rightarrow (\underbrace{a > 0}_{(i)} \wedge \underbrace{w \leq l}_{(ii)} \wedge \underbrace{p + c \geq m}_{(iii)})))$$

Property φ_2 verifies the consistency of orders that use coupons for free shipping. The premise of the implication lists the conditions for a completely specified order that uses such coupons. The conclusion checks the following business rules: (i) available quantity of the product is greater than zero, (ii) the weight of the product is in the limit allowed by the shipment method, and (iii) the total order value satifies the minimum for the application of the coupon.

We note that variants of LTL-FO have been introduced in [39, 70]. The use of globally quantified variables is also similar in spirit to the *freeze quantifier* defined in the context of LTL extensions with data by Demri and Lazić [28, 29].

Other applications of verification

As discussed in [30], various useful static analysis problems on business artifacts can be reduced to verification of temporal properties. We mention some of them.

Business rules The basic artifact model is extended in [8] with *business rules*, in order to support service reuse and customization. Business rules are conditions that can be super-imposed on the pre-conditions of existing services without changing their implementation. They are useful in practice when services are provided by autonomous third-parties, who typically strive for wide applicability and impose as unrestrictive pre-conditions as possible. When such third-party services are incorporated into a specific business process, this often requires more control over when services apply, in the form of more restrictive pre-conditions. Such additional control may also be needed to ensure compliance with business regulations formulated by third parties, independently of the specific application. Verification of properties in the presence of business rules then becomes of interest and can be addressed by our techniques. A related issue is the detection of *redundant* business rules, that do not affect the runs of the system. This can also be reduced to a verification problem.

Redundant attributes Another design simplification consists of redundant attribute removal, a problem also raised in [8]. This is formulated as follows. We would like to test whether there is a way to satisfy a property φ of runs without using one of the attributes. This easily reduces to a verification problem as well.

Runtime analysis The verification techniques described above can also be used to perform useful runtime analysis tasks. Examples include providing guidance to users trying to achieve certain goals, runtime monitoring of events, what-if scenarios, and diagnosis of anomalous behavior based on partial traces of an artifact execution (e.g. [2]). This is also in the spirit of the line of research on runtime operational support in BPM [71].

4. AUTOMATIC VERIFICATION OF ARTIFACT SYSTEMS

Classical model checking applies to finite-state transition systems. While finite-state systems may fully capture the semantics of some systems to be verified (for example logical circuits), most software systems are in fact infinite-state systems, of which a finite-state transition system represents a rough abstraction. Properties of the actual system are also abstracted, using a finite set of propositions whose truth values describe each of the finite states of the transition system. Checking that an LTL property holds is done by searching for a counterexample run of the system. Its finiteness is essential and allows to decide property satisfaction in PSPACE using an

automata-theoretic approach (see e.g. [22, 61]).

Consider now an artifact system $\mathcal{A}$ and an LTL-FO property φ. Model checking $\mathcal{A}$ with respect to φ can be viewed once again as a search for a counterexample run of $\mathcal{A}$, i.e. a run violating φ. The immediate difficulty, compared to the classical approach, stems from the fact that $\mathcal{T}_\mathcal{A}$ is an infinite-state system. To obtain decidability in this context, the typical approach consists of using *symbolic representations* of runs, as described later.

In the broader context of verification, research on automatic verification of infinite-state systems has also focused on extending classical model checking techniques (e.g., see [18] for a survey). However, in much of this work the emphasis is on studying recursive control rather than data, which is either ignored or finitely abstracted. More recent work has been focusing specifically on data as a source of infinity. This includes augmenting recursive procedures with integer parameters [13], rewriting systems with data [14, 12], Petri nets with data associated to tokens [54], automata and logics over infinite alphabets [16, 15, 64, 28, 52, 11, 12], and temporal logics manipulating data [28, 29]. However, the restricted use of data and the particular properties verified have limited applicability to the business artifact setting, or other database-driven applications.

Artifacts without constraints or dependencies

We consider first artifact systems and properties without arithmetic constraints or data dependencies. This case was studied in [30], with a slightly richer model in which artifacts can carry some limited relational state information (however, here we stick for simplicity to the earlier minimalistic model). The main result is the following.

THEOREM 4.1. *It is decidable, given an artifact system $\mathcal{A}$ with no data dependencies or arithmetic constraints, and an LTL-FO property φ with no arithmetic constraints, whether $\mathcal{A}$ satisfies φ.*

The complexity of verification is PSPACE-complete for fixed-arity database and artifacts, and EXPSPACE otherwise. This is the best one can expect, given that even very simple static analysis problems for finite-state systems are already PSPACE-complete [68].

The main idea behind the verification algorithm is to explore the space of runs of the artifact system using *symbolic* runs rather than actual runs. This is based on the fact that the relevant information at each instant is the pattern of connections in the database between attribute values of the current and successor artifact records in the run, referred to as their *isomorphism type*. Indeed, the sequence of isomorphism types in a run can be generated symbolically and is enough to determine satisfaction of the property. Since each isomorphism type can be represented by a polynomial number of tuples (for fixed arity), this yields PSPACE verification.

It turns out that the verification algorithm can be extended to specifications and properties that use a *total order* on the data domain, which is useful in many cases. This however complicates the algorithm considerably, since the order imposes global constraints that are not captured by the local isomorphism types. The algorithm was first extended in [30] for the case of a dense countable order with no end-points. This was later generalized to an arbitrary total order by Segoufin and Torunczyk [67] using automata-theoretic techniques. In both cases, the worst-case complexity remains PSPACE.

Artifacts with arithmetic constraints and data dependencies

Unfortunately, Theorem 4.1 fails even in the presence of simple data dependencies or arithmetic. Specifically, as shown in [30, 24], verification becomes undecidable as soon as the database is equipped with at least one key dependency, *or* if the specification of the artifact system uses simple arithmetic constraints allowing to increment and decrement by one the value of some atributes. Hence, a restriction is needed to achieve decidability. We discuss this next.

To gain some intuition, consider the undecidability of verification for artifact systems with increments and decrements. The proof of undecidability is based on the ability of such systems to simulate *counter machines*, for which the problem of state reachability is known to be undecidable [62]. To simulate counter machines, an artifact system uses an attribute for each counter. A service performs an increment (or decrement) operations by "feeding back" the incremented (or decremented) value into the next occurrence of the corresponding attribute. To simulate counters, this must be done an unbounded number of times. To prevent such computations, the restriction imposed in [24] is designed to limit the data flow between occurrences of the same artifact attribute at different times in runs of the system that satisfy the desired property. As a first cut, a possible restriction would prevent any data flow path between unequal occurrences of the same artifact attribute. Let us call this restriction *acyclicity*. While acyclicity would achieve the goal of rendering verification decidable, it is too strong for many practical situations. In our running ex-

ample, a customer can choose a shipping type and coupon and repeatedly change her mind and start over. Such repeated performance of a task is useful in many scenarios, but would be prohibited by acyclicity of the data flow. To this end, we define in [24] a more permissive restriction called *feedback freedom*. The formal definition considers, for each run, a graph capturing the data flow among variables, and imposes a restriction on the graph. Intuitively, paths among different occurrences of the same attribute are permitted, but only as long as each value of the attribute is independent on its previous values. This is ensured by a syntactic condition that takes into account both the artifact system and the property to be verified. We omit here the rather technical details. It is shown in [24] that feedback freedom of an artifact system together with an LTL-FO property can be checked in PSPACE by reduction to a test of emptiness of a two-way alternating finite-state automaton. More significantly, artifact systems designed in a hierarchical fashion by successive refinement, in the style of the Guard-Stage-Milestone approach [25, 47], naturally satisfy feedback freedom. Indeed, there is evidence that the feedback freedom condition is permissive enough to capture a wide class of applications of practical interest. This is confirmed by numerous examples of practical business processes modeled as artifact systems. Many of these, including typical e-commerce applications, satisfy the feedback freedom condition. Feedback freedom turns out to ensure decidability of verification in the presence of arithmetic constraints, and also under a large class of data dependencies including key and foreign key constraints on the database.

THEOREM 4.2. *[24] It is decidable, given an artifact system $\mathcal{A}$ whose database satisfies a set of key and foreign key constraints, and an LTL-FO property φ such that $(\mathcal{A}, \varphi)$ is feedback free, whether every run of $\mathcal{A}$ on a valid database satisfies φ.*

The intuition behind decidability is the following. Recall the verification algorithm of Theorem 4.1. Because of the data dependencies and arithmetic constraints, the isomorphism types of symbolic runs no longer suffice, because every artifact record in a run is constrained by the entire history leading up to it. This can be specified as an $\exists$FO formula using one quantified variable for each artifact attribute occurring in the history, referred to as the *inherited constraint* of the record. The key observation is that due to feedback freedom, the inherited constraint can be rewritten into an $\exists$FO formula with quantifier rank bounded by k^2, where k is the number of attributes of the artifact (the quantifier rank of a formula is the maximum number of quantifiers occurring along a path from root to leaf in the syntax tree of the formula, see [55]). This implies that there are only finitely many non-equivalent inherited constraints. This allows to use again a symbolic run approach to verification, by replacing isomorphism types with inherited constraints.

One might wonder if the positive results of this section can be extended to branching-time logics (CTL or CTL*). Unfortunately, it is easily shown that even very simple CTL properties become undecidable in the present framework. It remains open whether there are reasonable restrictions that guarantee decidability of CTL or CTL*. We note that limited positive results on verification of branching-time properties of data-driven web services are obtained in [34].

Other work on verification of artifact systems

Recently, a line of work on automatic verification of database-centric business processes (specified using formalisms isomorphic to artifact systems) has introduced a variant of the verification problem in which properties are checked only over the runs starting from a given initial database. During the run, the database may evolve via updates, insertions and deletions. In particular, it may be extended with fresh values provided as input throughout the run. Since inputs come from an infinite domain, this verification variant remains infinite-state. The property languages are fragments of first-order-extended μ-calculus [26]. Decidability results in this context are based on sufficient syntactic restrictions. One such restriction ensures that the number of fresh input values is bounded throughout every run [26, 46]. The restriction exploits an analogy between artifact system runs and sequences of chase steps with embedded dependencies, and it corresponds to the notion of "weakly acyclic" set of dependencies [40]. A complementary type of restriction allows an unbounded number of distinct inputs during the run, but not their unbounded accumulation within the database, implying a bound on the latter's size. This restriction, called "generate-recall acyclicity", is based on a data flow analysis of how cyclic generation of fresh inputs interacts with their cyclic storage (recall) during the run [46]. [5] derives decidability of the verification variant by also disallowing unbounded accumulation of input values, but this condition is postulated as a semantic property (shown undecidable in [46]).

Additional results on formal analysis of artifact-

centric business processes in restricted contexts have been reported in [43, 44, 8]. Properties investigated in these studies include reachability [43, 44], general temporal constraints [44], and the existence of complete execution or dead end [8]. Citations [43, 44] are focused on an essentially procedural version of artifact-centric workflow, and [8] is the first to study a declarative version. For the variants considered in each paper, verification is generally undecidable; decidability results were obtained when rather severe restrictions are placed, e.g., restricting all guards on state transitions to be "true" [43], restricting to bounded domains [44, 8], or restricting the language for conditions to refer only to artifacts (and not their attribute values) [44]. None of the above papers permits an arbitrary global database, separate from the artifacts. See [51] for a survey on data-centric business process management, and [19] for a survey of corresponding verification results.

5. DATA-DRIVEN WEB SERVICES

The goal of the Web services paradigm is to enable the use of Web-hosted services with a high degree of flexibility and reliability. Web services can function in a stand-alone manner, or they can be "glued" together into multi-peer *compositions* that implement complex applications. To describe and reason about Web services, various standards and models have been proposed, focusing on different levels of abstraction and targeting different aspects of the Web service. We refer to [50] for a tutorial.

We illustrate with an example the WebML approach to specifying data-driven web services, formally studied in [33, 35].

Consider the common scenario of a web service[2] that takes input from external users and responds by producing output. The contents of a Web page is determined dynamically by querying the underlying database as well as the state. The output of the Web site, transitions from one Web page to another, and state updates, are determined by the current input, state, and database, and defined by first-order queries. We are interested in services specified by a high-level tool such as WebML (and Web Ratio).

We illustrate in Figure 2 a WebML-style specification of an e-commerce Web site selling computers online. New customers can register a name and password, while returning customers can login, search for computers fulfilling certain criteria, add the results to a shopping cart, and finally buy the items in the shopping cart.

A run of the above Web site starts as follows. Customers begin at the home page by providing their login name and password, and choosing one of the provided buttons (login, register, or cancel). Suppose the choice is to login. The reaction of the Web site is determined by a query checking if the name and password provided are found in the database of registered users. If the answer is positive, the login is successful and the customer proceeds to the Customer page or the Administration page depending on his status. Otherwise, there is a transition to the Error page. This continues as described by the flowchart in the figure.

Verification of data-driven web services

The verification problem for database-driven web services has been studied using a transducer-based formal model, called *Extended Abstract State Machine Transducer*, in brief ASM$^+$. The transducer model captures in a simple way the essential features of relational database-driven reactive systems. The model is an extension of the Abstract State Machine (ASM) transducer previously studied by Spielmann [70]. Similarly to the earlier Relational Transducers of Abiteboul et. al. [3], ASM$^+$ transducers model database-driven reactive systems that respond to input events by producing some output, and maintain state information in designated relations. The control of the device is specified using first-order queries. The main motivation for ASM$^+$ transducers is that they are sufficiently powerful to simulate complex Web service specifications in the style of WebML. Thus, they are a convenient vehicle for developing the theoretical foundation for the verification of such systems, and they also provide the basis for the implementation of a verifier.

As in the case of business artifacts, restrictions are needed on the ASM$^+$ transducers and properties in order to ensure decidability of verification. The main restriction, first proposed in [70] for ASM transducers, is called "input boundedness". The core idea of input boundedness is that quantifications used in formulas of the specification and property are guarded by input atoms. For example, if *pay* is an input, the LTL-FO formula (where **B** is shorthand for *before*)

$$\forall x \ (\mathbf{G} \ (\exists z(pay(x,z) \wedge price(x,z)) \ \mathbf{B} \ ship(x)))$$

is input bounded, since the quantification $\exists z$ is guarded by $pay(x,z)$. This restriction matches naturally the intuition that the system modeled by the transducer is input driven. The actual restriction is quite technical, but provides an appealing package. First, it turns out to be tight, in the sense that

[2]We interpret "web service" broadly to include SOA-style services as well as web applications and web sites. Although artifact services are different, they may in fact be implemented by web services.

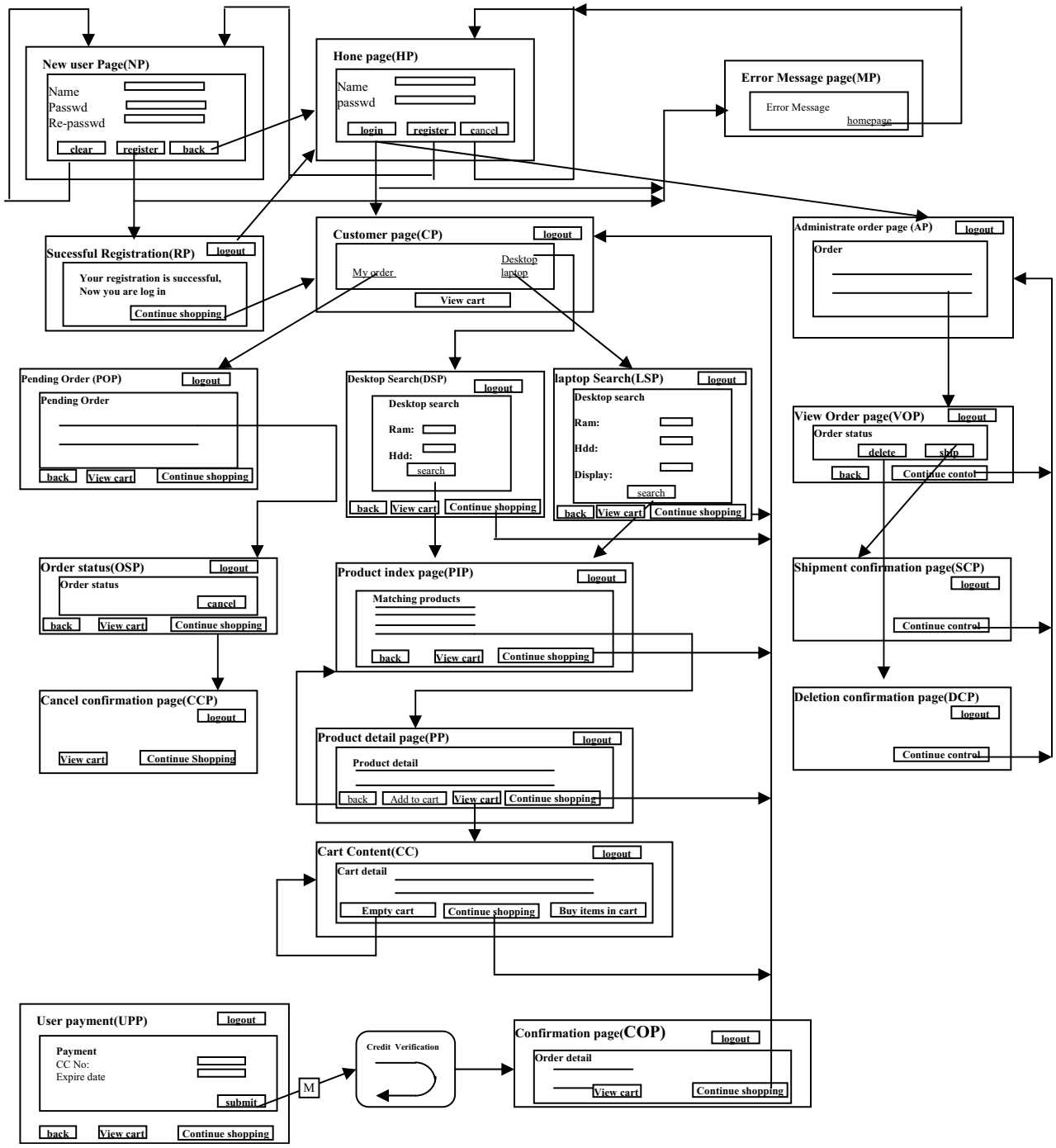

Figure 2: Web pages in the computer shopping site.

even small relaxations lead to undecidability. Second, as argued in [33, 35], it remains sufficiently rich to express a significant class of practically relevant applications and properties. As a typical example, the e-commerce Web application illustrated in Figure 2 can be modeled under this restriction, and many relevant natural properties can be expressed. Third, as in the case of artifacts without dependencies or arithmetic, the complexity of verification is PSPACE (for fixed-arity schemas). Moreover, the proof technique developed to show decidability in PSPACE provides the basis for the implementation of an actual verifier, described next.

The WAVE Verifier

While the PSPACE upper bound obtained for verification in the input-bounded case is encouraging from a theoretical viewpoint, it does not provide any indication of practical feasibility. Fortunately, it turns out that the symbolic approach described above also provides a good basis for efficient implementation. Indeed, this technique lies at the core of the WAVE verifier, targeted at data-driven Web services of the WebML flavor [37, 32].

The verifier, as well as its target specification framework, are both implemented from scratch. Thus, we first developed a tool for high-level, efficient specification of data-driven Web services, in the spirit of WebML. Next, we implemented WAVE taking as input a specification of a Web service using our tool, and an LTL-FO property to be verified. The starting point for the implementation is the symbolic run technique. Indeed, the verifier basically carries out a search for counterexample symbolic runs. However, verification becomes practical only in conjunction with an array of additional heuristics and optimization techniques, yielding critical improvements. Chief among these is dataflow analysis, allowing to dramatically prune the search for counterexample runs.

The verifier was evaluated on a set of practically significant Web application specifications, mimicking the core features of sites such as Dell, Expedia, and Barnes and Noble. The experimental results are quite exciting: we obtained surprisingly good verification times (on the order of seconds), suggesting that automatic verification is practically feasible for significant classes of properties and Web services. The implementation and experimental results are described in [32], and a demo of the WAVE prototype was presented in [36].

Compositions of ASM$^+$ Transducers

The verification results discussed above apply to single ASM$^+$ transducers in isolation. These results were extended in [37] to the more challenging but practically interesting case of *compositions* of ASM$^+$ transducers, modeling compositions of database-driven Web services. Asynchronous communication between transducers adds another dimension that has to be taken into account. In an ASM$^+$ composition, the transducers communicate with each other by sending and receiving messages via one-way channels. Properties of runs to be verified are specified in an extension of LTL-FO, where the FO components may additionally refer to the messages currently read and received.

Towards decidable verification, we extend in a natural way the input-boundedness restriction. Additional restrictions must be placed on the message channels: they may be lossy, but are required to be bounded. With these restrictions, verification is again shown to be PSPACE-complete (for fixed-arity relations, and EXPSPACE otherwise). The proof is by reduction to the single transducer case, and the restrictions are shown to be tight.

The above model of compositions assumes that all specifications of participating peers are available to the verifier. However, compositions may also involve autonomous parties unwilling to disclose the internal implementation details. In this case, the only information available is typically a specification of their input-output behavior. This leads to an investigation of *modular* verification. It consists in verifying that a subset of fully specified transducers behaves correctly, subject to input-output properties of the other transducers. Decidability results are obtained in [37] for verification, subject to an appropriate extension of the input-boundedness restriction.

6. CONCLUSIONS

Database-driven systems provide the backbone of many complex applications for which verification is critically important. A fortunate development facilitating this task is the emergence of high-level specification tools centered around database queries, that provide a natural target for verification. The results we described suggest that verification may indeed be feasible for significant classes of database-driven systems so specified. Specifically, we considered two representative models, business artifacts and data-driven web services, and identified restrictions guaranteeing decidability of verification. However, more work is needed at several levels in order to improve the applicability and im-

pact of the results.

On the theoretical front, the quest for the right package of restrictions that enable verification while capturing more relevant sets of specifications is still ongoing. For example, real-life artifacts often contain data beyond flat tuples, such as lists or sets (think of a shopping cart). While unrestricted use of sets is known to lead to undecidability of verification [30], preliminary results [31] suggest that carefully limited use of sets might preserve decidability, while allowing to model common use cases such as the shopping cart. Other useful extensions involve checking properties of the interaction among multiple actors in the workflow, or among multiple artifact instances evolving in parallel. The interoperation, evolution, and integration of multiple workflows also raise important static analysis questions.

Bridging the gap between the abstract setting of theoretical results and full-fledged specification frameworks also raises significant challenges. The decidability results are subject to strong restrictions. The boundary of decidability is subtle, as even small deviations from the restrictions may lead to undecidability. This raises a need to provide tools to guide the design of full-fledged specifications towards satisfaction of the restrictions, whenever possible (e.g. see [69]).

Clearly, a practical verifier needs to also deal with specifications that do not obey the restrictions needed for decidability. As typical in software verification, this can be done by abstracting the given specification to one that satisfies the restrictions, and verifying the resulting abstraction. For example, if certain arithmetic operations are not supported by the verifier, they can be abstracted as black-box relations, ignoring their semantics. The resulting verifier is guaranteed to be *sound* (it is never wrong when it claims correctness of a specification), but is possibly not *complete* (it may produce false negatives, i.e. candidate counterexamples to the desired property, which need to be validated by the user). The technical challenge lies in automatically generating the abstraction such that it gives up only as little completeness as necessary for decidability.

While the theoretical complexity results provide basic information on the difficulty of verification, its practical feasibility can only be demonstrated by actual implementations. The surprisingly good performance of the implemented WAVE verifier [32] is therefore particularly encouraging. Like the theoretical results, this is made possible by a novel coupling of database and model checking techniques. This suggests that the approach to verification described here is quite promising, and may be just the starting point of a fruitful marriage between the database and computer-aided verification areas.

7. REFERENCES

[1] Web Ratio. http://www.webratio.com/.

[2] S. Abiteboul and V. Vianu. Collaborative data-driven workflows: think global, act local. In *PODS*, 2013.

[3] S. Abiteboul, V. Vianu, B. Fordham, and Y. Yesha. Relational transducers for electronic commerce. *JCSS*, 61(2):236–269, 2000.

[4] B.B.Hariri, D.Calvanese, G. D. Giacomo, R. D. Masellis, and P.Felli. Foundations of relational artifacts verification. In *BPM*, 2011.

[5] F. Belardinelli, A. Lomuscio, and F. Patrizi. Verification of gsm-based artifact-centric systems through finite abstraction. In *ICSOC*, 2012.

[6] K. Bhattacharya, N. S. Caswell, S. Kumaran, A. Nigam, and F. Y. Wu. Artifact-centered operational modeling: Lessons from customer engagements. *IBM Sys. Journal*, 46(4), 2007.

[7] K. Bhattacharya et al. A model-driven approach to industrializing discovery processes in pharmaceutical research. *IBM Systems Journal*, 44(1), 2005.

[8] K. Bhattacharya, C. E. Gerede, R. Hull, R. Liu, and J. Su. Towards formal analysis of artifact-centric business process models. In *BPM*, 2007.

[9] BizAgi and Cordys and IBM and Oracle and SAP AG and Singularity (OMG Submitters) and Agile Enterprise Design and Stiftelsen SINTEF and TIBCO and Trisotech (Co-Authors). Case Management Model and Notation (CMMN), FTF Beta 1, Jan. 2013. OMG Document Number dtc/2013-01-01, Object Management Group.

[10] D. Boaz, L. Limonad, and M. Gupta. BizArtifact: Artifact-centric Business Process Management, June 2013. `http://sourceforge.net/projects/bizartifact/`.

[11] M. Bojanczyk, A. Muscholl, T. Schwentick, L. Segoufin, and C. David. Two-variable logic on words with data. In *LICS*, 2006.

[12] A. Bouajjani, P. Habermehl, Y. Jurski, and M. Sighireanu. Rewriting systems with data. In *FCT'07*.

[13] A. Bouajjani, P. Habermehl, and R. Mayr. Automatic verification of recursive procedures with one integer parameter. *Theoretical Computer Science*, 295:85–106, 2003.

[14] A. Bouajjani, Y. Jurski, and M. Sighireanu. A generic framework for reasoning about dynamic networks of infinite-state processes. In *TACAS'07*.
[15] P. Bouyer. A logical characterization of data languages. *Inf. Processing Letters*, 84(2), 2002.
[16] P. Bouyer, A. Petit, and D. Thérien. Algebraic approach to data languages and timed languages. *Inf. and Comp.*, 182(2), 2003.
[17] M. Brambilla, S. Ceri, S. Comai, P. Fraternali, and I. Manolescu. Specification and design of workflow-driven hypertexts. *Journal of Web Engineering*, 1(1), 2002.
[18] O. Burkart, D. Caucal, F. Moller, and B. Steffen. Verification of infinite structures. In *Handbook of Process Algebra*, pages 545–623. Elsevier Science, 2001.
[19] D. Calvanese, G. De Giacomo, and M. Montali. Foundations of data-aware process analysis: a database theory perspective. In *PODS*, 2013.
[20] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing data-intensive Web applications*. Morgan-Kaufmann, 2002.
[21] T. Chao et al. Artifact-based transformation of IBM Global Financing: A case study. In *BPM*, 2009.
[22] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 2000.
[23] D. Cohn, P. Dhoolia, F. Heath, F. Pinel, and J. Vergo. Siena: From powerpoint to web app in 5 minutes. In *ICSOC*, 2008.
[24] E. Damaggio, A. Deutsch, and V. Vianu. Artifact systems with data dependencies and arithmetic. In *ICDT*, 2011.
[25] E. Damaggio, R. Hull, and R. Vaculín. On the equivalence of incremental and fixpoint semantics for business artifacts with guard-stage-milestone lifecycles. *Information Systems*, 38:561–584, 2013.
[26] G. De Giacomo, R. D. Masellis, and R. Rosati. Verification of conjunctive artifact-centric services. *Int. J. Cooperative Inf. Syst.*, 21(2):111–140, 2012.
[27] H. de Man. Case management: Cordys approach. BP Trends (`www.bptrends.com`), 2009.
[28] S. Demri and R. Lazić. LTL with the Freeze Quantifier and Register Automata. In *LICS*, 2006.
[29] S. Demri, R. Lazić, and A. Sangnier. Model checking freeze LTL over one-counter automata. In *FoSSaCS*, 2008.
[30] A. Deutsch, R. Hull, F. Patrizi, and V. Vianu. Automatic verification of data-centric business processes. In *ICDT*, 2009.
[31] A. Deutsch, Y. Li, D. Lorant, and V. Vianu. Personal communication, 2014.
[32] A. Deutsch, M. Marcus, L. Sui, V. Vianu, and D. Zhou. A verifier for interactive, data-driven web applications. In *SIGMOD*, 2005.
[33] A. Deutsch, L. Sui, and V. Vianu. Specification and verification of data-driven web services. In *PODS*, 2004.
[34] A. Deutsch, L. Sui, and V. Vianu. Specification and verification of data-driven web applications. *JCSS*, 73(3):442–474, 2007.
[35] A. Deutsch, L. Sui, and V. Vianu. Specification and verification of data-driven web services. *JCSS*, 73(3):442–474, 2007.
[36] A. Deutsch, L. Sui, V. Vianu, and D. Zhou. A system for specification and verification of interactive, data-driven Web applications. In *SIGMOD*, 2006.
[37] A. Deutsch, L. Sui, V. Vianu, and D. Zhou. Verification of communicating data-driven Web services. In *PODS*, pages 90–99, 2006.
[38] G. Dong, R. Hull, B. Kumar, J. Su, and G. Zhou. A framework for optimizing distributed workflow executions. In *DBPL*, 1999.
[39] E. A. Emerson. Temporal and modal logic. In J. V. Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics*, pages 995–1072. North-Holland Pub. Co./MIT Press, 1990.
[40] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. In *ICDT*, 2003.
[41] M. F. Fernández, D. Florescu, A. Y. Levy, and D. Suciu. Declarative specification of web sites with Strudel. *VLDB Journal*, 9(1), 2000.
[42] D. Florescu, K. Yagoub, P. Valduriez, and V. Issarny. WEAVE: A data-intensive web site management system(software demonstration). In *EDBT*, 2000.
[43] C. E. Gerede, K. Bhattacharya, and J. Su. Static analysis of business artifact-centric operational models. In *SOCA*, 2007.
[44] C. E. Gerede and J. Su. Specification and verification of artifact behaviors in business process models. In *ICSOC*, 2007.
[45] R. Glushko and T. McGrath. *Document Engineering: Analyzing and Designing Documents for Business Informatics and Web Services*. MIT Press, Cmabridge, MA, 2005.
[46] B. B. Hariri, D. Calvanese, G. De Giacomo, A. Deutsch, and M. Montali. Verification of

relational data-centric dynamic systems with external services. In *PODS*, 2013.

[47] R. Hull, E. Damaggio, R. D. Masellis, F. Fournier, M. Gupta, F. H. III, S. Hobson, M. Linehan, S. Maradugu, A. Nigam, P. Sukaviriya, and R. Vaculín. Business artifacts with guard-stage-milestone lifecycles: Managing artifact interactions with conditions and events. In *ACM DEBS*, 2011.

[48] R. Hull, F. Llirbat, B. Kumar, G. Zhou, G. Dong, and J. Su. Optimization techniques for data-intensive decision flows. In *ICDE*, 2000.

[49] R. Hull, F. Llirbat, E. Simon, J. Su, G. Dong, B. Kumar, and G. Zhou. Declarative workflows that support easy modification and dynamic browsing. In *Proc. Int. Joint Conf. on Work Activities Coordination and Collaboration*, 1999.

[50] R. Hull and J. Su. Tools for design of composite web services. In *SIGMOD*, 2004.

[51] R. Hull, J. Su, and R. Vaculín. Data management perspectives on business process management: tutorial overview. In *SIGMOD*, 2013.

[52] M. Jurdzinski and R. Lazić. Alternation-free modal mu-calculus for data trees. In *LICS*, 2007.

[53] S. Kumaran, P. Nandi, T. Heath, K. Bhaskaran, and R. Das. ADoc-oriented programming. In *Symp. on Applications and the Internet (SAINT)*, 2003.

[54] R. Lazić, T. Newcomb, J. Ouaknine, A. Roscoe, and J. Worrell. Nets with tokens which carry data. In *ICATPN'07*.

[55] L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[56] R. Liu, K. Bhattacharya, and F. Y. Wu. Modeling business contexture and behavior using business artifacts. In *CAiSE*, 2007.

[57] M. Marin, R. Hull, and R. Vaculín. Data centric bpm and the emerging case management standard: A short survey. In *BPM Workshops*, 2012.

[58] D. Martin et al. OWL-S: Semantic markup for web services, W3C Member Submission, November 2003. `http://www.daml.org/services/`.

[59] S. A. McIlraith, T. C. Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, 16(2):46–53, 2001.

[60] G. Mecca, P. Merialdo, and P. Atzeni. Araneus in the era of XML. *IEEE Data Engineering Bulletin*, 22(3):19–26, 1999.

[61] S. Merz. Model checking: a tutorial overview. In *Modeling and verification of parallel processes*. Springer-Verlag New York, 2001.

[62] M. L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, 1967.

[63] P. Nandi and S. Kumaran. Adaptive business objects – a new component model for business integration. In *Proc. Intl. Conf. on Enterprise Information Systems*, 2005.

[64] F. Neven, T. Schwentick, and V. Vianu. Finite State Machines for Strings Over Infinite Alphabets. *ACM Transactions on Computational Logic*, 5(3):403–435, 2004.

[65] A. Nigam and N. S. Caswell. Business artifacts: An approach to operational specification. *IBM Systems Journal*, 42(3), 2003.

[66] A. Pnueli. The temporal logic of programs. In *FOCS*, 1977.

[67] L. Segoufin and S. Torunczyk. Automata based verification over linearly ordered data domains. In *STACS*, 2011.

[68] A. Sistla and E. Clarke. The complexity of propositional linear temporal logic. *J. of the ACM*, 32:733–749, 1985.

[69] D. Solomakhin, M. Montali, S. Tessaris, and R. D. Masellis. Verification of artifact-centric systems: Decidability and modeling issues. In *ICSOC*, 2013.

[70] M. Spielmann. Verification of relational transducers for electronic commerce. *JCSS.*, 66(1):40–65, 2003.

[71] W. M. P. van der Aalst. *Process Mining*. Springer, 2011.

[72] J. Wang and A. Kumar. A framework for document-driven workflow systems. In *BPM*, 2005.

[73] W.-D. Zhu et al. Advanced Case Management with IBM Case Manager. Available at `http://www.redbooks.ibm.com/abstracts/sg247929.html?Open`.

Towards Total Traffic Awareness

Chenjuan Guo[†] Christian S. Jensen[‡] Bin Yang[†]

[†] Department of Computer Science, Aarhus University, Denmark

[‡] Department of Computer Science, Aalborg University, Denmark

[†] {cguo, byang}@cs.au.dk, [‡] csj@cs.aau.dk

ABSTRACT

A combination of factors render the transportation sector a highly desirable area for data management research. The transportation sector receives substantial investments and is of high societal interest across the globe. Since there is limited room for new roads, smarter use of the existing infrastructure is of essence. The combination of the continued proliferation of sensors and mobile devices with the drive towards open data will result in rapidly increasing volumes of data becoming available. The data management community is well positioned to contribute to building a smarter transportation infrastructure. We believe that efficient management and effective analysis of big transportation data will enable us to extract transportation knowledge, which will bring significant and diverse benefits to society. We describe the data, present key challenges related to the extraction of *thorough*, *timely*, and *trustworthy* traffic knowledge to achieve total traffic awareness, and we outline services that may be enabled. It is thus our hope that the paper will inspire data management researchers to address some of the many challenges in the transportation area.

1. INTRODUCTION

Transportation adversely affects many people's daily lives, and increasingly so. For example, as cities continue to grow, congestion gets worse and affects more and more people. And emissions from vehicles are responsible for high concentrations of airborne particles that increasingly threaten the health of people. For example, air pollution is believed to have contributed to as many as 1.2 million and 600,000 premature deaths in China and India in 2010[1]. Emissions also contribute to the greenhouse effect associated with the accelerating global warming that threatens to considerably affect the conditions for life on Earth.

Rapidly growing volumes of data that captures the state of a transportation infrastructure are becoming available, due to several developments. Infrastructure users increasingly carry mobile devices capable of contributing data. The infrastructure is increasingly being instrumented with data acquisition devices, e.g., infrared counting devices, Bluetooth and Wi-Fi base stations that "see" mobile devices, and cameras. As the movement towards open public data continues, it is a safe bet that such data will become available in large volumes.

An important and challenging goal is to use this data to achieve total traffic awareness. Individual data sources generally cover only a limited part of a transportation infrastructure. To enable global awareness of an entire infrastructure, the integration, or fusion, of multiple and diverse data sources is essential. To enable up-to-date awareness, new query processing techniques are called for that are capable of ingesting rapid streams of data, reflecting the data in query results with near-zero latency. The resulting total traffic awareness enables improved as well as new applications and services.

The availability of very interested stakeholders and large volumes of data allows empirical evaluations of the feasibility, effectiveness, and efficiency of data management proposals.

We note that the transportation setting is markedly different from that of the so-called *smart dust*, which was proposed in the beginning of the 1990's and gained substantial attention in the early 2000's. A key idea was to disperse large quantities of tiny sensing and communication devices in some environment, e.g., a rain forest, in order to monitor that environment. The devices would organize into a wireless sensor network that would stream data to users. While this is a compelling vision, the sizes of deployments are tiny. We are (luckily) unaware of any rain forests having been littered by large quantities of devices. In contrast, the "sensor network" of transportation is already up and running, and the continued operation of cities increasingly depend on the effective use of the data being generated.

We proceed to characterize available transportation data in Section 2. Then we describe challenges inherent in achieving total traffic awareness in Section 3, and

[1] http://tinyurl.com/bl48fq2

Types	Techniques	Accuracy	Cost	Dynamic Properties	Objects
Individual	CANBus	High	Low	Fuel consumption	Moving, stationary
	GNSS	High	Low	Instantaneous velocities, latitude-longitude locations	
Collective	PS	Low	Low	Travel times, average velocities, traffic flow	Stationary
	Cameras	High	High		
	LDs	High	High		
Media	LBSNs	Low	Low	Instant events, scheduled events, weather conditions	Moving, stationary
	Radio	High	Low		
	Web	High	Low		

Table 1: Dynamic Data Gathering Techniques

we outline applications and services that this enables in Section 4.

2. TRANSPORTATION DATA

Transportation data describes properties of stationary and moving objects that influence travel. Stationary objects include elements of a transportation infrastructure, e.g., road segments, intersections, points of interest (POIs), and regions of interest (ROIs). Moving objects includes vehicles, pedestrians, and location-based social network (LBSN) users. Both types of objects have *static* and *dynamic* properties.

Stationary objects have static properties, which may be described in *static data* sources. For example, the length, speed limit, and toll cost of a road segment may be recorded in digital maps and web pages of road authorities. The management of such data calls for spatial-data integration, e.g., location entity matching [1] and geospatial data fusion [2]. Moving objects also have static properties, such as, sizes, weights, or engine types of vehicles.

Dynamic properties of stationary and moving objects are described by *dynamic data* that can be gathered in different ways. For example, an important dynamic property of a road segment is its time-dependent travel time distribution across a day and a week, which can be obtained from data collected by Bluetooth and Wi-Fi sensors installed along the roads. We categorize three types of dynamic data gathering techniques in Table 1.

Individual gathering techniques capture individual moving objects' dynamic properties. A controller area network bus (CANBus) is an in-vehicle network that connects sensors that measure a variety of vehicle-related data, e.g., fuel consumption, at some frequency. A Global Navigation Satellite System (GNSS), e.g., GPS or Galileo, is able to capture a moving object's instantaneous velocities and latitude-longitude locations at some frequency (up to every 0.02 seconds[2]). The accuracies of the reported properties are high. For example, fuel consumption can be recorded with error rates between -2% and +2%[3].

The data collected by individual moving objects also relates to different stationary objects. Thus, it is possible to infer travel times or fuel consumptions associated with the traversals of road segments.

Collective gathering techniques are deployed at fixed locations in a transportation network, and they excel at capturing dynamic properties of stationary objects, e.g., the traffic flow of a particular road segment. Presence sensing (PS) techniques (e.g., Bluetooth, Wi-Fi, RFID, and Infrared), cameras, and loop detectors (LDs) are able to detect *occurrences* of moving objects at fixed locations where devices are deployed. Given the distance between, e.g., two cameras, and timestamps of vehicles' occurrences, the average speeds of road segment can be derived.

These techniques vary according to the fraction of objects they are able to detect. Cameras and LDs are able to capture almost all moving objects that pass by, while PS techniques are only able to capture some 20 to 30%[4] of the objects that pass by. However, deployment and maintenance costs of PS techniques are relatively low.

Media gathering techniques rely on humans to contribute data, e.g., about an accident, a scheduled event such as a football match, or the weather. These techniques generally report on dynamic properties of stationary and moving objects.

Location-Based Social Networks (LBSNs) are good at capturing instant and scheduled events, e.g., in the form of check-ins and user-generated content (*UGGC*) such as geo-tagged tweets and photos. The density of captured events depends on the density of users and on how frequently they post on LBSNs. The accuracy can vary greatly.

Radio stations can broadcast data on instant and sched-

[2] http://tinyurl.com/m7wqerc

[3] http://tinyurl.com/op6trtx

[4] http://tinyurl.com/low8vg8

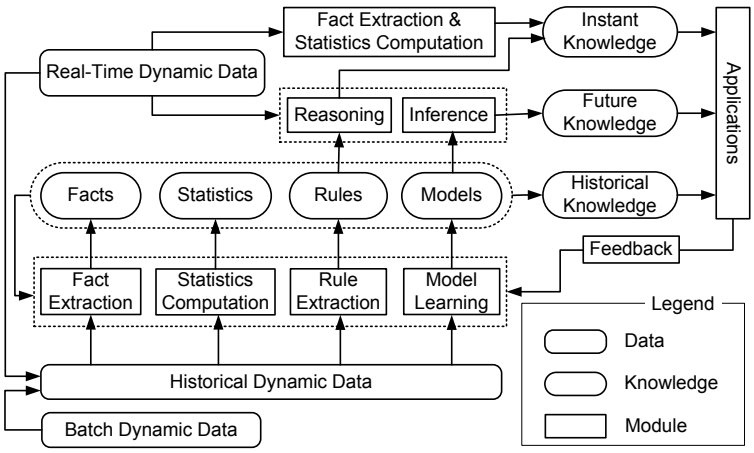

Figure 1: From Transportation Data to Thorough, Timely, and Trustworthy Transportation Knowledge

uled events and weather conditions. The accuracy is relatively high, and the deployment cost is relatively low. The web is also a good resource of schedules events and weather conditions, and accuracy is high and cost is low.

Finally, dynamic data can be classified into *real-time*, *batch*, and *historical* data. *Real-time* data is delivered immediately after being collected, *batch* data is accumulated and then delivered in batches according to some protocol, and *historical data* is delivered some time after it is collected.

3. TOTAL TRAFFIC AWARENESS

To support different applications and services, we envision a system that transforms data into transportation knowledge; see the architecture in Figure 1.

3.1 Knowledge Representation and Properties

Transportation knowledge takes the forms of *facts*, *statistics*, *rules*, and *models*. We assume that transportation data is cleaned and pre-processed before being passed through the modules in Figure 1.

Facts describe stationary and moving objects' dynamic properties, such as the average velocity of a vehicle passing through a segment. Facts are the product of integration and consolidation of dynamic data across different data sources. Thus, the *fact extraction* module integrate data elements that describe the same property of an object but are stored in different sources. It is often appropriate to associate facts with an estimated reliability.

Statistics are temporal and spatial aggregations on dynamic data and facts, such as the distribution of travel times with which vehicles pass through a road segment or the frequent routes that a user traverses between home and work during April. The *statistics computation* module must ensure the integrity of data and facts while describing them more compactly.

Rules make it possible to infer stationary-object properties from the properties of moving objects, and vice versa. For example, if at least $x\%$ of detected vehicles having velocities smaller than y then a road segment is considered as congested. And if a road segment is congested, the time needed to traverse it exceeds z minutes. Rules are produced by the *rule extraction* module.

Models describe stochastic processes that represent possible evolutions of objects across time. For example, given the current vehicle density at a road segment, a model may predict possible densities at the segment and adjacent segments 15 minutes into the future. Given an object's past trajectories, a model may predict the the object's future trajectories. The *model learning* module derives such models that are subsequently fed with real-time dynamic data to infer possible dynamic, future properties.

Next, transportation knowledge can be classified according to its temporal aspect. **Historical knowledge** describes past traffic and is obtained from the historical transportation data. **Instant knowledge** describes current or near-past traffic. It is derived by means of *fact extraction*, *statistics computation*, and *reasoning* from streaming real-time, dynamic data. Examples include the current vehicle density of a road segment and the congestion status of a road segment. **Future knowledge** infers future traffic, e.g., whether a segment is congested or clear 15 minutes from now. This knowledge is obtained from models and real-time dynamic data.

Three aspects are essential to achieve total traffic awareness. First, *thoroughness* relates to the spatial, temporal, and property coverage. Spatial coverage is thorough if the knowledge covers an entire transportation infrastructure; temporal coverage is thorough if the knowledge covers an entire period of interest; and property coverage is thorough if the knowledge fully covers the

traffic properties of interest (e.g., travel times, fuel consumption). Next, *timeliness* implies that the available knowledge is up-to-date. For example, a self-driving car requires knowledge as to whether it can go across an intersection to be at most milliseconds old, while a few seconds of delay may be acceptable for a continuous routing service. Third, *trustworthiness* means that the knowledge is sufficiently accurate and reliable for its use, even if it is derived from inaccurate and uncertain data. The knowledge should come with a quantification of its accuracy and reliability.

3.2 Challenges

To achieve total traffic awareness, it is necessary to effectively and efficiently utilize the available static and dynamic transportation data. Table 2 offers an overview of key challenges.

Thoroughness: Any single type of dynamic data is unable to offer thoroughness by itself. For example, camera data can only cover the locations where cameras are deployed. Thus, data integration is of essence. Traditional techniques, e.g., schema alignment [3] and data fusion [4], need to be adapted to the spatial-temporal aspects. Further, integration must contend with the general characteristics of transportation data. Recent techniques for *big data integration* [5,6] can be helpful. Major challenges relate to sparsity and duplicate detection.

Sparsity: Although integrating various types of dynamic data increases thoroughness, *spatial*, *temporal*, and *property* data sparsity must be addressed.

Spatial sparsity occurs because some roads lack sufficient data. For example, no data is available for a road with no PS, LDs, or camera deployment if no vehicle with a contributing GNSS devices has traversed it. Borrowing data from nearby and topologically similar roads [7, 8] may be useful for obtaining knowledge for such roads.

Extrapolation-related techniques may be used for addressing temporal sparsity. Knowledge for a road during a period when no data is available can be inferred from data from nearby periods [9], from data from nearby roads that have data for the relevant period [8], or from data from nearby or similar roads with data from nearby periods [10, 11].

Property sparsity occurs when no data captures a desired property. In such cases, it may be possible to exploit related data. For example, if only GNSS data is available for a road segment, but fuel consumption is desired, it is possible to feed the GNSS data to environmental impact models to derive fuel consumption data [12].

To fully contend with sparsity, major challenges remain. Many existing methods rely on complex mathematical optimizations that do not scale to big transportation data. Scalable solutions to solving complex optimizations are missing. Alternatively, novel problem formulations that exploit scalable techniques, e.g., scalable matrix operations, are needed.

Second, existing methods often assume static scenarios rarely consider real-time data. In contrast, our setting calls for techniques that are able to adapt to real-time data.

Third, existing techniques consider the three sparsity aspects individually, and typically rely on one type of data (primarily GNSS data). Techniques that are able to consider all the three sparsity aspects and to exploit multiple data sources in a holistic manner are called for.

Duplicate detection: If a moving object is detected by more than one data gathering technique, duplicate records are generated. For example, assume that we want to know the number of vehicles passing through a road segment during a short period. GNSS records may suggest 5, while PS records may suggest 7. Simply adding 5 and 7 is wrong if one vehicle is detected by both techniques.

Duplicate detection aims to identify the data that describes same moving objects, where the data is collected by different techniques. A simple heuristic for identifying duplicates is that if trajectories provided by different techniques are highly consistent, they may refer to the same moving object. However, the heterogeneity of trajectories is not addressed well in most of existing trajectory clustering methods [13]. Recent advances in duplicate detection in dynamic settings [14] may offer a good starting point, but cannot be applied directly.

Timeliness: Ensuring up-to-date traffic awareness presents several challenges.

Incremental maintenance: Historical knowledge, e.g., a model for predicting the travel time on a segment, is built from historical data. As such data accumulates, the historical knowledge may change. When and how the historical knowledge would change are usually unknown and cannot be predicted. This calls for an efficient and effective approach to incrementally update and maintain historical knowledge.

While we expect that it will be relatively easy to contend with facts and rules, the real challenge lies in how to maintain models. One possibility is to adopt an online learning approach, where models are updated frequently using very recent data. However, this approach may be sensitive to unscheduled traffic events, such as accidents or road construction. Another strategy may be to update the historical knowledge only when recent data disagrees significantly with the existing historical knowledge. To summarize, incremental maintenance challenges include: (1) how to efficiently and effectively identify the (dis)agreement between the existing historical knowledge and recent data; (2) how to efficiently

Properties	Challenges	Historical Knowledge	Instant Knowledge	Future Knowledge
Thoroughness	Dealing with sparsity	✓	✓	✓
	Duplicate detection	✓	✓	✓
Timeliness	Incremental maintenance	✓		
	Efficient retrieval	✓		
	Efficient processing		✓	✓
Trustworthiness	Conflict reconciliation	✓	✓	
	Veracity enhancement	✓	✓	
	Accurate prediction			✓

Table 2: Total Traffic Awareness Challenges

and effectively differentiate the two cases; and (3) how to address disagreements.

Efficient retrieval: We consider the efficient retrieval of historical knowledge. While some techniques for efficient historical knowledge retrieval do exist, we face the specific challenge that our knowledge is spatio-temporal and takes four forms.

There is a need for efficient retrieval that involves comparisons between historical knowledge and streaming real-time data. For instance, when an accident happens, it is of interest to predict the spatio-temporal extent of the congestion in the road network that is caused by the of the accident. Facts and statistics from a similar past accident that happened in a similar situation (e.g., same region, at a similar time of day, under similar weather condition) may provide reliable predictions. Rules (e.g., if an intersection is congested, its X% adjacent segments and its Y% 2-nd adjacent segments will be congested) and models (e.g., prediction of the duration of congestion) at the current accident location may also help predict the impact of the accident. How to efficiently retrieve historical facts, statistics, rules, and models that are relevant to given dynamic real-time data is an important challenge.

Efficient processing: It should be possible to generate instant and future knowledge efficiently from historical knowledge and dynamic real-time data so that up-to-date knowledge is available to applications.

The generation of facts, statistics, and traffic statuses from rules must contend with the specifics of transportation data and therefore faces challenges akin to this covered for real-time data integration [15]. As the data sources that provide transportation data keep changing, additional challenges result. Also, the solutions to the thoroughness problems should also be addressed efficiently.

Using models to predict future traffic is another challenge. Some proposals, e.g., [10, 16], address this problem. However, whether these proposals are scalable and work in a rea-time manner is unknown.

Trustworthy: Different types of data have different veracity. For example, camera and loop detectors capture all or nearly all vehicles, while Bluetooth and Wi-Fi base stations do not. Our setting is faced with large amounts of low-veracity data. However, the extracted transportation knowledge should be trustworthy. This causes a number of challenges, including the following.

Conflict reconciliation: Data from different sources may disagree on the same property of a moving object. For example, GNSS data may suggest that a vehicle travels at 50 km/h, while PS data may suggest 55 km/h. The challenge is how to derive a single, trustworthy value for a property given conflicting data, and how to quantify the trustworthiness.

Methods considering data source trustworthiness (e.g., weighted voting) [17, 18] can be adopted. Here, each type of data is associated with a weight reflecting how trustworthy it is. Using the weights, a single trustworthy value for a property can be determined. The key is how to determine the weights. Sometimes, the weight of a technique may vary due to, e.g., weather conditions. A method that can automatically assign appropriate weights and update weights when necessary is highly desired. A possible solution is to use high-veracity data, e.g., camera data, as training data, and to assign and update the weights of other types.

Veracity enhancement: Veracity enhancement aims to extract high-veracity knowledge from low-veracity data from multiple sources. An interesting solution may be to utilize social knowledge (e.g., social media data that mentions traffic, or crowdsourcing) to help increase veracity [19]. A challenge is how to utilize social knowledge in an on-line setting.

Accurate prediction: It is challenging to use historical model knowledge to accurately predict near-future traffic. This may call for novel types of models capable of accommodating traffic dynamics and that are robust enough to deal with low veracity data, e.g., by automat-

ically dropping outlier data.

4. APPLICATIONS

Total traffic awareness enables a range of applications and services, a few of which we review here.

Routing: Given a source-destination pair, a routing service suggests routes. *Eco-routing* provides routes that minimizes greenhouse gas emissions. Eco-routing needs historical knowledge to construct time-varying eco-weights that describe emissions on road segments across time and may also need instant and future knowledge to update eco-weights. Concerns for travel time and distance may also be integrated into eco-routing, yielding *multi-criteria routing* [20]. Next, *continuous-routing* utilizes current and future knowledge to provide up-to-date routes, e.g., the fastest route, from a driver's current location to the driver's destination as traffic conditions change. Finally, in *context aware, personalized routing*, different drivers are provided with different routes that best match their current preferences.

Parking: The objective is to help drivers find parking. *Capacity notification* uses current and future knowledge of parking availability to help drivers. *Nearby parking* suggests the nearest available parking. This requires historical knowledge of parking spaces that are not recorded in digital maps (e.g., on-street parking [21]) and also needs instant knowledge of current availability.

Event Response: This relates to how to respond to an event, e.g., a traffic accident or a football match. *Event detection* concerns the discovery of events. A scheduled event, e.g., a football match, can be identified from, e.g., web pages or social media. An unscheduled event, e.g., an accident, needs to be detected from instant knowledge. *Event effect* predicts the spatio-temporal effect of an event from historical, instant, and future knowledge. *Event notification* aims to notify travelers of relevant events in advance. This calls for comparison of a traveler's movement with the extent of an event. This requires instant and future knowledge.

5. REFERENCES

[1] V. Sehgal, L. Getoor, and P. Viechnicki, "Entity resolution in geospatial data integration," in *GIS*, pp. 83–90, 2006.

[2] S. Stankutė and H. Asche, "An integrative approach to geospatial data fusion," in *ICCSA*, pp. 490–504, 2009.

[3] P. A. Bernstein, J. Madhavan, and E. Rahm, "Generic schema matching, ten years later," *PVLDB*, 4(11): 695–701, 2011.

[4] X. L. Dong and F. Naumann, "Data fusion - resolving data conflicts for integration," *PVLDB*, 2(2): 1654–1655, 2009.

[5] S. Guo, X. Dong, D. Srivastava, and R. Zajac, "Record linkage with uniqueness constraints and erroneous values," *PVLDB*, 3(1): 417–428, 2010.

[6] X. Liu, X. L. Dong, B. C. Ooi, and D. Srivastava, "Online data fusion," *PVLDB*, 4(11): 932–943, 2011.

[7] T. Idé and M. Sugiyama, "Trajectory regression on road networks," in *AAAI*, 2011.

[8] B. Yang, M. Kaul, and C. S. Jensen, "Using incomplete information for complete weight annotation of road networks," *TKDE*, 2014.

[9] J. Zheng and L. M. Ni, "Time-dependent trajectory regression on road networks via multi-task learning," in *AAAI*, 2013.

[10] B. Yang, C. Guo, and C. S. Jensen, "Travel cost inference from sparse, spatio-temporally correlated time series using markov models," *PVLDB*, 6(9): 769–780, 2013.

[11] Y. Zhu, Z. Li, H. Zhu, M. Li, and Q. Zhang, "A Compressive Sensing Approach to Urban Traffic Estimation with Probe Vehicles," *IEEE Trans. Mob. Comput.*, 12(11): 2289–2302, 2013.

[12] C. Guo, Y. Ma, B. Yang, C. S. Jensen, and M. Kaul, "Ecomark: evaluating models of vehicular environmental impact," in *SIGSPATIAL/GIS*, pp. 269–278, 2012.

[13] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory clustering: a partition-and-group framework," in *SIGMOD*, pp. 593–604, 2007.

[14] P. Li, X. L. Dong, A. Maurino, and D. Srivastava, "Linking temporal records," *PVLDB*, vol. 4, no. 11, pp. 956–967, 2011.

[15] C. Rueda and M. Gertz, "Real-time integration of geospatial raster and point data streams," in *SSDBM*, pp. 605–611, 2008.

[16] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *KDD*, pp. 316–324, 2011.

[17] A. Galland, S. Abiteboul, A. Marian, and P. Senellart, "Corroborating information from disagreeing views," in *WSDM*, pp. 131–140, 2010.

[18] X. Yin, J. Han, and P. S. Yu, "Truth discovery with multiple conflicting information providers on the web," *TKDE*, 20(6): 796–808, 2008.

[19] A. Artikis, "Heterogeneous stream processing and crowdsourcing for urban traffic management," in *EDBT*, pp. 712–723, 2014.

[20] B. Yang, C. Guo, C. S. Jensen, M. Kaul, and S. Shang, "Stochastic Skyline Route Planning Under Time-Varying Uncertainty," in *ICDE*, pp. 136–147, 2014.

[21] B. Yang, N. Fantini, and C. S. Jensen, "iPark: identifying parking spaces from trajectories," *EDBT*, pp. 705–708, 2013.

A Survey on XML Fragmentation

Vanessa Braganholo
Fluminense Federal University, UFF
Brazil
vanessa@ic.uff.br

Marta Mattoso
Federal University of Rio de Janeiro, COPPE/UFRJ
Brazil
marta@cos.ufrj.br

ABSTRACT

Efficient document processing is a must when large volumes of XML data are involved. In such critical scenarios, a well-known solution to this problem is to distribute (map) the data among several processing nodes, and then distribute the processing accordingly, taking advantage of parallelism. This is the approach taken by distributed databases and MapReduce environments. Fragmentation techniques play an important role in these scenarios. They provide a way to "cut" the database into pieces and distribute the pieces over a network. This way, queries can also be "cut" into sub-queries that run in parallel, thus achieving better performance when compared to the centralized environment. However, there is no consensus in the database community as to what an XML fragment is. In fact, several approaches in literature present definitions of XML fragments. In addition to query processing, using XML fragmentation techniques may also be helpful when managing XML documents distributed along the web or clouds. This paper surveys the existing XML fragmentation approaches in literature, comparing their features and highlighting their drawbacks. Our contribution resides in establishing a map of the area.

1. INTRODUCTION

Efficient document processing is a must when large volumes of XML data are involved [35, 56]. To achieve good performance in query processing, lots of initiatives focus on indexing [13, 14, 18, 20, 21, 34, 57] and query optimization [15, 16, 29, 55, 74]. In addition to these initiatives, distributed query processing can further improve the performance when large collections of documents are involved. In critical analytical scenarios, complex queries, such as OLAP (On-Line Analytical Processing), cannot effectively benefit from indexing techniques since queries involve several attributes and are *ad-hoc*.

MapReduce [22, 23] and Hadoop[1] have been extensively used to execute operations in parallel based on *ad-hoc* sub-sets of data. Originally defined for searching web log datasets, MapReduce is being progressively used for other types of datasets. Horizontal data fragmentation, allocation and indexing techniques [26] have been proposed to improve Hadoop's performance and experiments have been performed over a variety of data, including TPC-H OLAP queries [71]. Other types of fragmentation have also been used to process large XML datasets using Hadoop [17]. In fact, MapReduce has been compared [24, 25, 61, 67] and combined [3, 71] with Parallel Databases approaches, always aiming at achieving better response times for query processing. Such strategies can be used to process large XML datasets if XML documents are correctly fragmented and reconstructed.

The term XML *fragment* appeared in the beginning of the last decade denoting a well-formed piece of an XML document [69, 60] (in fact, the first draft related to the W3C candidate recommendation appeared in June of 1999). The main goal at that time was to ease the communication between applications, so they could exchange pieces of documents instead of entire ones. After that, the term XML fragment was used (with several different definitions) in the context of distributed databases [2, 12, 47, 64], and distributed query processing issues began to raise researchers' interest. In essence, these definitions can be summarized as follows: an XML fragment is a (not necessarily well-formed) piece of an XML document or subset of an XML collection.

Fragmentation techniques provide a way to "cut" the database into pieces (fragments). This way, queries can also be "cut" into sub-queries that run in parallel over smaller portions of data, thus achieving better performance when compared to its serial execution over the whole document. Successful parallel query processing in relational databases has been directly achieved through table fragmentation. By using the relational algebra to define

[1] `http://hadoop.apache.org`

fragments, algebraic query processing on these fragments can be done with correctness rules [58]. However, there is no consensus in the database community as to what an XML fragment precisely is. In fact, several approaches in literature present definitions of XML fragments [4, 8, 11, 12, 39, 42, 41, 43, 47, 62]. They can be classified according to the way they fragment the data. Regardless of the fragmentation type, the fragmentation unit is usually a collection of elements, which can be of Multiple Documents (MD) or Single Document (SD) [75].

Given the large amount of approaches that propose fragmentation alternatives in literature, in this paper we survey the existing approaches, comparing their features and establishing a timeline, so the reader can understand the evolution of the proposed solutions, their characteristics, pros and cons. It is important to note that, when we refer to distribution, we are not considering distributed query processing in data integration scenarios [5, 33, 44, 68], since their main goal is to provide data access, not necessarily in a high performance fashion. We also do not consider approaches that fragments the document solely for internal query processing [40, 53, 72]. Instead, we focus on approaches that aim to improve the performance on processing collections of documents by distributing data to several nodes in a network and use parallel processing strategies such as MapReduce.

The remaining of this paper is organized as follows. Section 2 provides intuitive definitions for XML fragmentation, and Section 3 formalizes them. Section 4 discusses fragmentation techniques, while Section 5 discusses their main features, comparing them regarding several aspects. Finally, we discuss open problems in Section 6.

2. AN INTUITIVE GRASP ON XML FRAGMENTATION

Data fragmentation is characterized by physical changes to the dataset, that is, the dataset is fragmented and allocated to multiple computational nodes [4, 8, 12, 39, 42, 41, 43, 47]. In their book [58], Ozsu and Valduriez present a detailed introduction to the subject of XML distributed query processing and XML fragmentation. They classify fragmentation in two groups: *ad-hoc* and structured. *Ad-hoc* fragmentation does not take the document schema into consideration. Edges are arbitrarily removed from the document. Structured fragmentation, on the other hand, is one that uses schema property(ies) to define the fragments [58]. In this work we are focused on large-scale distributed and parallel query processing. In structured fragmentation, each fragment is usually defined by using

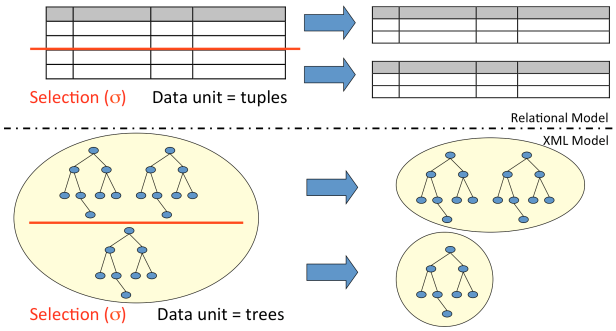

Figure 1: Horizontal fragmentation in the relational model and in the XML model

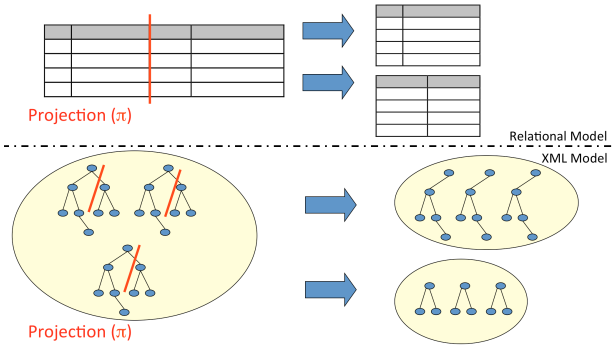

Figure 2: Vertical fragmentation in the relational model and in the XML model

set operations (i.e. algebraic selection, projection) over the dataset. In this case, the specification of queries and fragments share the same operations, which makes it easier to automatically decompose queries to run in parallel over the fragments.

To help the intuitive grasp on XML fragments we refer to fragmentation in the relational model [58], where horizontal fragments are those obtained by table selection operations. This means that horizontal fragments are restricted by the selection predicate and follow the same schema of the original table. Vertical fragments, on the other hand, are defined by projection operations, and thus follow a different schema of the original table. The same principle can be applied to XML databases [4, 42]. Figure 1 and Figure 2 establish a parallel between data fragmentation in the relational and in the XML models.

In distributed databases, queries are executed over one specific target fragment or in parallel by accessing different fragments of the dataset independently. Data fragmentation can be very effective in distributed query processing where there is a well-known set of frequent queries. Such queries must be analyzed so that a corresponding fragmentation design[2] can be achieved [6, 8, 42, 48]. A

[2]The process of deciding on how to fragment the

good fragmentation design is one that benefits the frequent queries, thus allowing for gains in performance. Gains are focused on locality of access and data pruning rather than parallel processing. The price to be paid is poor performance for some of the non-frequent queries and limited opportunities for high performance parallel queries.

To illustrate, assume an XML collection *COrders* that contains information about customer's orders. Queries are usually targeted at regions as follows: South America, North America or Other Continent. Assume also that the collection is physically fragmented into three fragments, based on their location: one containing orders of customers from North America (allocated at node *n1*), one containing orders of customers from South America (allocated at node *n2*), and a third one containing orders from other Continents (allocated at node *n3*).

Now, assume that customers from North America buy items usually over 1,000 dollars. On the other hand, South American customers' orders are usually around 500 dollars. Now, suppose we want to run a query to retrieve the average total of orders with items above 1,000. This query will be distributed to the three fragments: sub-query *s1* will be executed over the North America fragment, sub-query *s2* will query the South America fragment, and sub-query *s3* will run over the remaining fragment. Since there are a lot more orders from customers in North America that satisfy this query predicate, *s2* and *s3* will probably finish their processing way before *s1*. Thus, the total query processing time will be highly influenced by *s1,* and there is nothing *n2* and *n3* can do to help *n1* processing *s1*, since they do not have the required data. If there is a following operation for these intermediate results, skew continues to be propagated. Even if the fragments are replicated, since the sub-query *s1* is already running in *n1* over the whole North America fragment there is no way to stop this subquery (assuming query processing follows a static optimized execution plan) and redistribute the elements of this long processing fragment to idle nodes.

This type of fragmentation favors the distributed processing with the goal of restricting the access to a subset of the data by using pruning strategies. This happens in cases where a query accesses a subset of the fragments (orders from North America with items above 2,000 dollars, for instance). When the goal is high performance, horizontal and vertical fragmentation can still be used. The idea in this case is to generate uniform fragments, with similar number of tuples/elements each. The goal is to favor parallel processing by using the largest possible number of processing units. Still in this case, data skew is a problem, since the query processing time will also be highly influenced by selectivity factors and the node that contains the largest number of tuples/elements that satisfy the query predicate can be overloaded.

database (how each fragment should be defined) is called *fragmentation design.*

3. FORMAL DEFINITIONS

Based on the intuition of XML fragmentation presented in the previous section, we now formalize the main concepts related to XML fragmentation. As mentioned in the introduction, there is no consensus in the literature as to what an XML fragment exactly is. In this section, we present the definitions proposed in [4] as an illustrative example.

XML Document. XML documents consist of trees with nodes labeled by element names, attribute names or constant values. Let $\mathcal{L}$ be the set of distinct element names, $\mathcal{A}$ the set of distinct attribute names, and $\mathcal{D}$ the set of distinct data values. An *XML data tree* is denoted by the expression $\Delta := \langle t, \ell, \Psi \rangle$, where: t is a finite ordered tree, ℓ is a function that labels nodes in t with symbols in $\mathcal{L} \cup \mathcal{A}$; and Ψ maps leaf nodes in t to values in $\mathcal{D}$. The root node of Δ is denoted by $root_\Delta$. We assume nodes in Δ do not have mixed content; if a given node v is mapped into $\mathcal{D}$, then v does not have siblings in Δ. Notice, however, that this is not a limitation, but rather a presentation simplification. Furthermore, nodes with labels in $\mathcal{A}$ have a single child whose label must be in $\mathcal{D}$. An XML document is a data tree.

Types. Basically, names of XML elements correspond to names of data types, described in a DTD or XML Schema. Let S be a schema. We say that document $\Delta := \langle t, \ell, \Psi \rangle$ *satisfies* a type τ, where $\tau \in S$, iff $\langle t, \ell \rangle$ is a tree derived from the grammar defined by S such that $\ell(root_\Delta) \rightarrow \tau$. A *collection* C of XML documents is a set of data trees. We say it is *homogeneous* if all the documents in C satisfy the same XML type. If not, we say the collection is *heterogeneous.* Given a schema S, a homogeneous collection C is denoted by the expression $C := \langle S, \tau_{root} \rangle$, where τ_{root} is a type in S and all instances Δ of C satisfy τ_{root}. Figure 3 shows the *SOrders* schema that was used in the example given in Section 2. Over this schema, we can define collections such as $COrders := \langle SOrders, order \rangle$, or $CItems := \langle SOrders, items \rangle$, both of Multiple Documents.

Path Expression. A *path expression* P is a sequence $/\mathtt{e}_1/\ldots/\{\mathtt{e}_k \mid \mathtt{@a}_k\}$, where $\mathtt{e}_x \in \mathcal{L}$, $1 \leq x \leq k$,

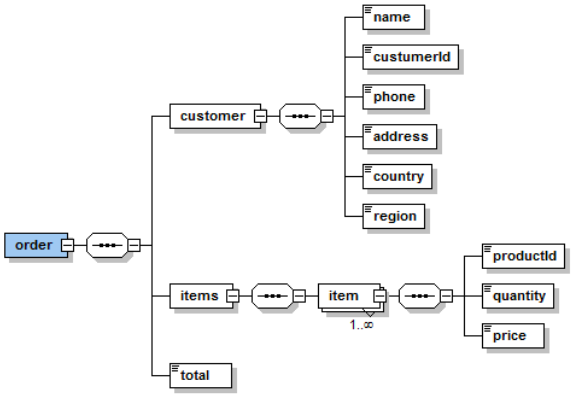

Figure 3: *SOrders* sample schema

and $a_k \in \mathcal{A}$. P may optionally contain "$*$" to indicate any element, and "//" to indicate any sequence of descendant elements. Besides, the term $e[i]$ may be used to denote the i-th occurrence of element e. The evaluation of a path expression P in a document Δ represents the selection of all nodes with label e_k (or a_k) whose steps from $root_\Delta$ satisfy P. P is said to be *terminal* if the content of the selected nodes is simple (i.e., if they have domain in $\mathcal{D}$). On the other hand, a *simple predicate* p is a logical expression: $p := P\,\theta\,value \mid \phi_v(P)\,\theta\,value \mid \phi_b(P) \mid Q$, where P is a terminal path expression, $\theta \in \{=, <, >, \neq, \leq, \geq\}$, $value \in \mathcal{D}$, ϕ_v is a function that returns values in $\mathcal{D}$, ϕ_b is a boolean function and Q denotes an arbitrary path expression. In the latter case, p is true if there are nodes selected by Q (existential test).

The following XML fragmentation definition builds on the semantics of the operators from the TLC (Tree Logical Classes) algebra [59], since it is one of the few XML algebras [30, 32, 37, 63, 76, 77] that operates on sets of data trees, which are natural operands for set fragmentation operations.

XML Fragment. A fragment F of a homogeneous collection C is a collection represented by $F := \langle C, \gamma \rangle$, where γ denotes an operator defined over C. F is horizontal if γ denotes a selection; vertical, if operator γ is a projection; or hybrid, when there is a composition of select and project.

Instances of a fragment F are obtained by applying γ to each document in C. The collection of the resulting documents form the fragment F, which is valid if all documents generated by γ are well-formed (i.e., they must have a single root).

Horizontal Fragmentation. A horizontal fragment F of a collection C is defined by the selection operator (σ) [59] applied over documents in C, where the predicate of σ is a boolean expression with one or more simple predicates. Thus, F has the same schema of C.

$$FCustomer := \langle COrders, \pi_{/order,\{/order/items\}} \rangle$$
$$FItems := \langle COrders, \pi_{/order/items,\{\}} \rangle$$

Figure 5: Example of vertical fragments

Let μ be a conjunction of simple predicates over a collection C. The horizontal fragment of C defined by μ is given by the expression $F := \langle C, \sigma_\mu \rangle$, where σ_μ denotes the selection of documents in C that satisfy μ, that is, F contains documents of C for which μ is true. Figure 4 shows the definition of the horizontal fragments we used in Section 2. Notice that the third fragment is defined as the complement of the other two.

Notice that, by definition, SD repositories may not be horizontally fragmented, since horizontal fragmentation is defined over trees (instead of nodes). However, the elements in an SD repository may be distributed over fragments using a hybrid fragmentation, as described later.

Vertical Fragmentation. A vertical fragment is obtained by applying the projection operator (π) [59] to "split" a data structure into smaller parts that are frequently accessed in queries. Observe that, in XML repositories, the projection operator has a quite sophisticated semantics: it is possible to specify projections that exclude subtrees whose root is located in any level of an XML tree. A projection over a collection C retrieves, in each document of C (notice that C may have a single document, in case it is of type SD), a set of subtrees represented by a path expression, which are possibly pruned in some descendant nodes.

Let P be a path expression over collection C. Let $\Gamma := \{E_1, \ldots, E_x\}$ be a (possibly empty) set of path expressions contained in P (that is, path expressions in which P is a prefix). A vertical fragment of C defined by P is denoted $F := \langle C, \pi_{P,\Gamma} \rangle$, where $\pi_{P,\Gamma}$ denotes the projection of the subtrees rooted by nodes selected by P, excluding from the result the nodes selected by the expressions in Γ. The set Γ is called the prune criterion of F. As an example, we could separate customers from items, placing them in different fragments, as shown in Figure 5. Note that this is an alternative fragmentation, non-related to the one shown in Figure 4.

It is worth mentioning that the path expression P cannot retrieve nodes that may have cardinality greater than one, except when the element order is indicated. This restriction assures that the fragmentation results in well-formed documents, without the need of generating artificial elements to reorganize the subtrees projected in a fragment.

Hybrid Fragmentation. The idea of hybrid fragmentation is to apply a vertical fragmentation followed by a horizontal fragmentation, or vice-versa.

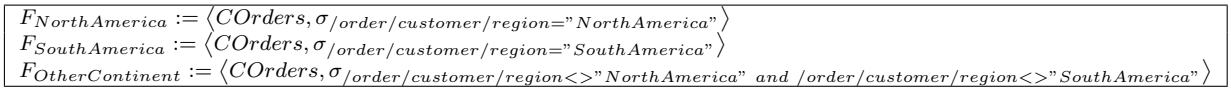

Figure 4: Example of horizontal fragments

An interesting use of this technique is to normalize the schema of XML collections in SD repositories, thereby allowing horizontal fragmentation.

Let σ_μ and $\pi_{P,\Gamma}$ be selection and projection operators, respectively, defined over a collection C. A hybrid fragment of C is denoted by $F := \langle C, \pi_{P,\Gamma} \bullet \sigma_\mu \rangle$, where $\pi_{P,\Gamma} \bullet \sigma_\mu$ denotes the selection of the subtrees projected by $\pi_{P,\Gamma}$ that satisfy μ.

Correctness Rules. Consider that a collection C is decomposed into a set of fragments $\Phi := \{F_1, ..., F_n\}$. The following rules must be verified to guarantee the correct fragmentation of C:

Completeness: each data item in C must appear in at least one fragment $F_i \in \Phi$. In the horizontal fragmentation, the data item consists of an XML document, while in the vertical fragmentation, it is a node.

Disjointness: for each data item d in C, if $d \in F_i$, $F_i \in \Phi$, then d cannot be in any other fragment $F_j \in \Phi$, $j \neq i$.

Reconstruction: it must be possible to define an operator ∇ such that $C := \nabla F_i$, $\forall F_i \in \Phi$, where ∇ depends on the type of fragmentation. For horizontal fragmentation, the union ($\cup$) operator [37] is used (TLC is an extension of TAX [37]), and for vertical fragmentation, the join ($\bowtie$) operator [59] is used.

These rules are important to guarantee that queries are correctly translated from the centralized environment to the corresponding fragmented one, and that results are correctly reconstructed.

4. FRAGMENTATION TECHNIQUES

In the next subsection we discuss *ad-hoc* fragmentation alternatives and then we focus on several approaches for structured fragmentation, following Ozsu and Valduriez [58] fragmentation classification.

4.1 Ad-hoc Fragmentation

Ad-hoc fragmentation approaches do not take the schema into consideration when defining the fragments. To generate the fragments, they either arbitrarily cut the document and mark it in a way that it can be reconstructed later, or use some kind of constraint over the document. We call these alternatives holes and fillers, and constraint-based, respectively.

Holes and Fillers. Bose et al propose a fragmentation model for stream data [11]. Using this fragmentation model, Bose and Fegaras [10] focus on processing fragmented data streams using a system they call XFrag. In this context, the goal is to fragment stream data and send it through the network. XFrag uses the concept of holes and fillers to fragment XML documents that will be sent over the network as data streams. The original document is divided into several smaller documents called fillers. Each fragment may contain one or more holes, where other fragments (the fillers) may fit. Such holes are marked with special tags *stream:hole*, which reference the ID of its corresponding filler. The information about the structure of the original document is called *tag structure*. Roughly speaking, it describes the DTD of the XML document that is to be fragmented and assigns an *id* to each element type.

Lee, Kim and Kang [45] analyze the classical holes and fillers approaches for stream query processing [10, 36] and claim they are inefficient in terms of memory consumption. To overcome this limitation, they propose to use a labeling scheme to connect vertical fragments of an XML document. In a XML tree, a labeling schema (such as global order or Dewey encoding [70]) can be used to connect parent and child. In the same way, Lee, Kim and Kang use a labeling schema to connect vertical fragments, where each fragment is a subtree of the original tree. Despite the ad-hoc nature of this approach, the authors mention the importance of the reconstruction property for correctness. Although there are not an explicit notion of holes and fillers in this approach, each fragment carries an id that is used to connect it to its parent, which is similar to the other holes and fillers approaches.

The query processing technique is also different from the ones applied to stored data. If we were to apply traditional distributed query processing techniques with the fragmentation schema proposed in [11], query processing would be inefficient, since it would be necessary to completely reconstruct the document before processing the query. This is because XFrag does not distinguish between horizontal, vertical and hybrid fragmentation. Also, it is not possible to describe fragments using predicates, which makes it impossible to define horizontal or hybrid fragments.

The *ad-hoc* fragmentation proposed by Abiteboul et al. [1, 2] use remote function calls (web ser-

vices) to fragment XML documents. The approach is called Active XML, and the goal is to guarantee that documents are up-to-date with respect to the ever-augmenting dynamic issues in current distributed and parallel computing environments. From time to time, or at query time, the web services are called and the results are embedded as XML fragments into the Active XML document. Thus, in this approach, web service calls represent cross-fragment edges [58]. Note that this can be seen as a holes and fillers approach, where web service calls are considered holes, which mark fragmentation spots, and their results are considered fillers.

Constraint-based. Bonifati and Cuzzocrea [8] propose a fragmentation approach based on structural constraints of the XML document: size, tree-width, and tree-depth. Given values for these three constraints, their approach finds ways of fragmenting the original tree to satisfy the constraints. The approach is based on a set of heuristics, and it is called SimpleX. There is no separation of fragmentation types, and no correctness rules. The approach was designed to work with stream data, but can be applied over stored data as well. The problem is that there are several possible ways of fragmenting a database while respecting the constraints. In this sense, Waldvogel, Kramis and Graf [73] propose five split algorithms and evaluate their performance, with the aim at finding the one that produces the most effective fragments.

Choi et al [17] use MapReduce to process a set of small XPath queries in parallel over large XML documents in an approach called HadoopXML. The input XML file is fragmented so that data blocks of equal size are produced. This is similar to the size constraint proposed by Bonifati and Cuzzocrea [8], and, as such, there is also no correctness rules or a formal concept of fragment type. The main goal of HadoopXML is to process as many small queries as possible in parallel. High-cost *ad-hoc* queries are not addressed.

4.2 Structured Fragmentation

According to Ozsu and Valduriez, *ad-hoc* fragmentation works well when data is already distributed. However, since there is no clear fragmentation predicate, there is less opportunity for distributed query optimization. An alternative that addresses this issue is structured fragmentation, which is based on the concept of fragmenting an XML data collection according to some properties of the schema [58].

Structured fragmentation can be performed in several ways. In this survey, we classify the existing approaches according to the way they define the fragments, which can be: hybrid, XPath-based, and set-oriented.

Hybrid. Ma and Schewe [47] base their XML fragments definition on ideas from fragmentation of object databases. In fact, a previous publication of Schewe contrasts fragmentation for these two models [64]. In their work, Ma and Schewe propose three types of XML fragmentation: *horizontal*, which groups elements of an XML document according to some selection criteria; *vertical*, which restructures a document by unnesting some elements; and a special type named *split*, that breaks an XML document into a set of new documents. Despite the use of the names *horizontal* and *vertical*, their fragments are not purely based on selection and projection. For example, horizontal fragmentation involves data restructuring and elements projection, thus yielding fragments with different schema definitions. Also, vertical fragmentation requires the specification of artificial elements to restructure the document, and artificial attributes to properly connect document fragments (for reconstruction purposes). Even though the user can define fragments from several XML documents, their approach is not suitable for MD repositories. In this case, to define horizontal fragments, the user must first integrate all XML documents into an SD view. It is important to note that they followed up with their work [49, 50] by proposing a cost model to help the fragments design aiming at reducing the query processing time [48, 52, 51].

XPath-based. Bremer and Gertz [12] propose an approach for distributed XML design, covering both data fragmentation and allocation. Fragments are defined using XF, a subset of XPath. Each fragment definition consists of two parts: a *selection fragment* and optionally a set of *exclusion fragments*. The selection fragment is applied to the XML database, resulting in a set of nodes N. Then, the exclusion fragments are applied over N. The results are the desired fragments. It is clear that this formalism does not distinguish between horizontal and vertical fragmentation, which are combined into a hybrid type of fragment definition. Nevertheless, their approach only addresses SD repositories. They aim to maximize local query evaluation by replicating global information, and distributing some index structures. They present important performance improvements, but their empirical evaluation focuses on the benefits of such indexes. Although Bremer and Gertz mention correctness rules, they do not present a reconstruction rule, which is crucial for automatic query processing.

Bonifati et al. [9] also define vertical fragments using XPath expressions. In the fragments definition, path expressions may contain child axes and positional filters. One fragment may possibly reference many fragments: a single super fragment, that is an ancestor of the current fragment, and (possibly many) child fragments. Child fragments are connected to their parent through *sub* tags that are artificially inserted into the parent fragment. Each *sub* tag references a child fragment. These definitions are used in a DHT (distributed hash table) P2P (peer-to-peer) system that supports XPath lookup queries. Since no selection predicate is allowed in the fragment definition, this approach does not consider horizontal nor hybrid fragmentation. An extended version of this paper, including experimental results, appeared in a journal paper in 2006 [7]. The main goal of the experiment was to measure the number of hops needed to find a query answer in the DHT, assuming the data is fragmented according to their definitions. Note that this approach has also the idea of holes and fillers, but it is classified as structured since it uses XPath to define the fragments.

The approach of Jeong et al. [38] also resembles the idea of holes and fillers. It splits the XML tree into subtrees with the goal of easing the processing of keyword-based queries. Fragments are defined using XPath and connected in a tree of fragments that is called Fragment Object Tree. There is no clear distinction between fragment types, but it is possible to use filters in the XPath expressions to define fragments (which plays the role of selections), and each of the several XPath expressions that define the Fragment Object Tree plays the role of a projection. Thus, we could classify this approach as providing a hybrid fragmentation.

Fegaras et al. [29] propose a declarative language to specify MapReduce jobs over XML documents called MRQL [28]. They propose a fragmentation technique based on the Hadoop input format. Fragments are defined by synchronization nodes and XPath expressions that are applied over these nodes. There are no correctness rules, but there is an algebra behind MRQL that is used to perform query optimization.

Set-oriented. Andrade et al. [4] defined horizontal, vertical and hybrid XML fragments inspired by the analogous definitions for the relational model [58]. Their approach is called PartiX and explores the analogy between relations and collections of trees (both are *sets*). It supports both SD and MD databases. Fragments are defined by XML algebra expressions [59]. A horizontal fragment is defined by a selection operation, while a vertical fragment is defined by a projection, plus an optional set of path expressions that point to subtrees to be pruned out of the fragment. A hybrid fragment is defined by a selection followed by a projection, or vice-versa. The definitions presented on Section 4 are extracted from this work, which is pioneer in the sense of formally defining correctness rules for the fragmentation definition. As in the relational model, the same algebra is used to define fragments and query predicates, which helps query decomposition and predicate matching. By using the same algebra and correctness rules, they were able to define a query processing methodology that is capable of automatically processing queries over distributed and fragmented databases [59].

Kido, Amagasa and Kitagawa [39] proposed horizontal and vertical fragmentation for XML data that was also inspired by the relational model. In their work, they use Data Guides [34] as the schema definition for the XML database. Their fragments are then specified over the Data Guide, which can be represented as a graph. A vertical fragment is a sub-graph of the graph that represents the database schema. A horizontal fragment is defined over a vertical fragment. When compared to the approach of Andrade et al. [4], the horizontal fragment of Kido, Amagasa and Kitagawa [39] is similar to the hybrid fragment of Andrade et al. They do not have a specific horizontal fragment as presented in PartiX [4], and no reconstruction rule either.

Kling, Ozsu and Daudjee [41] propose vertical fragmentation for XML databases. Their definition of vertical fragments is similar to that of [39]. However, they do not define horizontal nor hybrid fragments, although they mention they should be defined by selections (horizontal fragments) and selection plus projections or vice-versa (hybrid fragments).

This, in fact, is done in their following work [42], where Kling, Ozsu and Daudjee present definitions for horizontal and vertical fragments. The revised ideas in [42] are equivalent to those of Andrade et al. [4]. A horizontal fragmentation algorithm is defined over MD collections by using selection predicates and minterm[3] combinations, resulting in homogeneous fragments. Vertical fragments are defined by partitioning the schema of the documents into disjoint subgraphs, and can be applied to either SD or MD collections. Finally, horizontal and vertical fragments can be combined into hybrid fragments. The focus of their fragmentation design technique, however, is on pruning fragments for distributed

[3]A minterm is a conjunction of simple predicates [58].

query processing rather than on high performance parallel processing. It is targeted on a previously known frequent set of queries.

A common problem to [4] and [42] is that the use of minterms lead to a fixed number of fragments, disregarding the number of available processing units. This approach applies to distributed databases, but not to parallel query processing, since the number of fragments is typically much smaller than the current available processing units, which leads to idle processors and load unbalance. Since the number of processing units can be very dynamic, any design with a fixed number of fragments will require additional techniques to provide for high performance.

5. DISCUSSION

Now that we have discussed the main approaches on physical XML fragmentation, in this section we study the impact they had in literature by establishing a timeline. Finally, we analyze the features of each of the proposed approaches, summarizing their differences. Our goal is to try to obtain a uniform view on the XML fragments definitions and show the benefits of the different fragmentation design techniques so the designer can choose according to the target query-processing scenario.

In the timeline, shown in Figure 6, we analyze when each of the approaches was proposed. Ad-hoc approaches are shown in green, while structured approaches are shown in blue. White rectangles denote derived approaches. We consider a paper to be derived from a previous one if it discusses new features based on previous definitions. For instance, Figueiredo, Braganholo and Mattoso [31] used the definition of Andrade et al. [4] to propose a methodology for distributed XML query processing, and thus it is shown in white in the Figure. For derived approaches, we use a solid line to point to the original approach (either a green or blue rectangle). This way we can analyze the impact and continuity of each of the individual approaches. The dotted line that connects Ma and Schewe's approach with Schewe's approach denotes a previous work that pavements the subject, but do not present the fragments definitions themselves.

Additionally, each approach is positioned into a lane, according to its classification as constraint-based, holes and fillers, XPath-based, set-oriented or hybrid. This helps one to have a better understanding of how each type of approach evolved over time. While the timeline evidences that the subject has been continuously investigated throughout the years, Table 1 and Table 2 summarize the features of each of the approaches for structured (Table 1) and *ad-hoc* (Table 2) fragmentation, shown in Figure 6.

The first interesting feature to note is that, from all of the approaches, only two support fragmentation of MD collections [4, 42]. These two approaches are equivalent in terms of ideas. The only difference resides in the formalism that is used to present them. Ma and Schewe [47] also mention support to MD collections, but this needs to be done by first defining a single SD view, and then applying the fragmentation over this SD view. This maneuver could be applied to the other approaches as well, but it creates an artificial layer that needs to be handled by the user (that needs to create the views), and also at query processing time (it may decrease query performance). All of the approaches support fragmentation of SD collections.

Regarding the fragmentation types, most of the approaches allow a hybrid type of fragment [4, 12, 28, 38, 39, 42, 47]. In fact, in [39], a horizontal fragment is a subset of instances conforming to a vertical fragment. Thus, to be able to achieve horizontal fragmentation, one needs first to vertically fragment the database. Due to that reason, we consider this to be a hybrid fragment instead of a horizontal one. In a similar line of thought, Ma and Schewe [47] define their horizontal fragments by using selection and projection operations. In this paper, we consider them to be hybrid fragments, since horizontal fragments are classically defined only by selection operations [58]. However, we classified both [47] and [39] as allowing horizontal fragments in Table 1, preserving the classification given by the paper authors. According to this discussion and reclassification of fragmentation types, only [4, 42] support pure horizontal fragmentation.

As for vertical fragments, four out of nine structured approaches [4, 39, 42, 41] support pure vertical fragmentation. Ma and Schewe allow a special type of projection that unnests elements and arrange then into new artificial elements in the XML document. Thus, they do not use pure projections to define the vertical fragments. Additionally, they require artificial attributes to be created to support document reconstruction. Such attributes contain references to connecting fragments. Similarly, Bonifati et al. require an artificial element called sub to connect vertical fragments. Thus, we do not consider it to use pure projections.

Since ad-hoc fragmentation does not rely on an explicit fragmentation specification, Table 2 does not present the fragmentation types.

As mentioned before, in the same way as in the relational model [10], correctness rules are needed for

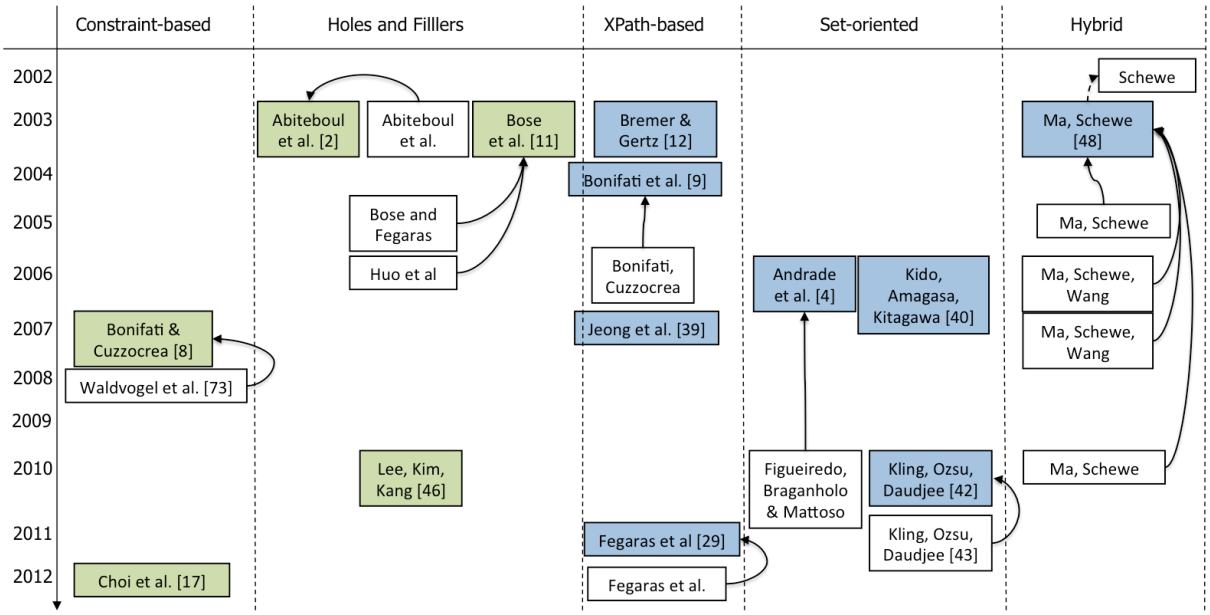

Figure 6: Timeline of the XML fragmentation approaches

automatic distributed query processing. When we use the same XML algebraic operations in fragmentation definition and query processing, automatic mappings from centralized to distributed fragments can be achieved following on correctness rules. Ma and Schewe [47] mention the need of creating artificial attributes for reconstruction purposes. However, only three of the proposed approaches present or mention correctness rules [4, 12, 39]. The approach in [39], however, does not present them formally, and does not include the reconstruction rule, which is crucial for reconstructing the original database from its fragments and vice-versa, thus allowing a correct mapping from centralized database to the corresponding distributed database. This allows for automatic distributed query processing on top of the mapped fragments. Bremer and Gertz's approach [12], on the other hand, does mention the importance of a reconstruction rule, but does not define it. Only [4] formally defines the three correctness rules. In fact, these rules are then explored in their following work [59] to automatically process distributed queries over the fragmented database.

Finally, most of the approaches deal with stored data [2, 4, 8, 12, 28, 38, 39, 42, 41, 47] instead of streams, and use native XML databases to store the data [2, 4, 8, 12, 38, 42, 41, 47]. Approaches based on MapReduce use HDFS to store the data [17, 28].

6. OPEN PROBLEMS

Based on our analysis of previous work in XML fragmentation, we have identified a list of open problems, which we discuss next.

Fragmentation Design. When fragmentation design is used, the most frequent queries must be known to derive a good fragmentation schema - one that benefits the most frequent queries. Experiments in literature show that queries that do not benefit from the fragmentation design suffer large impacts on performance [39, 59]. Only a few work in literature present algorithms for XML fragmentation design [6, 8, 42, 48, 66]. However, these algorithms were not used in parallel query processing and they do not present correctness rules for the resulting fragmentation. This makes it difficult to check the correctness of the fragmentation schema designed by the algorithms.

Even in cases where fragmentation design was performed, it cannot be easily adapted to changes in the query processing environment (for example, the addition of processing nodes). Additionally, pruning irrelevant fragments limits parallelism and performance gains, since several computational nodes remain idle when they have fragments that are irrelevant to the query that is being processed. In fact, experimental results [42] show that the pruning algorithm does not improve query performance.

Implementation. The absence of correctness rules also affects query processing. Most of the approaches show evaluation performance of query processing, and thus they implemented a prototype that is capable of processing queries. However, due to the absence of correctness rules, their prototype becomes a black box, very complex to be re-implemented by people outside their research group. This also prevents these approaches from

Table 1: Summary of the approaches for structured fragmentation

	MD	SD	horizontal	vertical	hybrid	correctness rules	source data	storage
Ma, Schewe [47]	N	Y	selection + projection	artificial elements	–	–	stored	native
Bremer, Gertz [12]	N	Y	–	–	selection fragments + exclusion fragments	disjointness, completeness	stored	native
Bonifati et al. [9]	N	Y	–	projection and artificial elements	–	–	stored	native
Andrade et al. [4]	Y	Y	selection	projection	selection + projection	disjointness, completeness, reconstruction	stored	native
Kido, Amagasa, Kitagawa [39]	N	Y	subset of the vertical fragments	subgraph of a dataGuide	–	disjointness, completeness	stored	relations
Jeong et al. [38]	N	Y	–	–	selection + projection	–	stored	native
Kling, Ozsu, Daudjee [41]	N	Y	–	subset of a schema graph	–	–	stored	native
Kling, Ozsu, Daudjee [42]	Y	Y	selection	projection	selection + projection	–	stored	native
Fegaras et al. [29]	N	Y	–	–	selection + projection	–	stored	HDFS

Table 2: Summary of the approaches for ad-hoc fragmentation

	MD	SD	correctness rules	source data	storage
Abiteboul et at. [2]	N	Y	–	stored or stream	native
Bose at al. [11]	N	Y	–	stream	–
Bonifati, Cuzzocrea [8]	N	Y	–	stored or stream	native
Lee, Kim, Kang [45]	N	Y	–	stream	–
Choi et al. [17]	N	Y	–	stored	HDFS

being used in Map-Reduce settings. We have unsuccessfully searched for publicly available implementations of each of the approaches, but only found one of Fegaras et al. [28]. In fact, it is currently an incubated project at Apache, which is available at `http://lambda.uta.edu/mrql`. As for the ad-hoc approaches, only Active XML is publicly available at the OW2 open source portal (`http://forge.ow2.org/projects/activexml`).

Virtual Fragmentation. The approaches we discuss in this paper can all be classified as physical fragmentation, since they physically break the XML tree into several pieces. Virtual fragmentation [54] is an alternative to physical fragmentation, and consists of replicating the database into several nodes and distributing the query into subqueries so that each node runs over a different portion of the data. Virtual fragmentation approaches [62] do not suffer from problems related to the physical fragmentation design, since data is replicated in all nodes, and queries run over all available nodes, each over a small non-overlapping portion of the data. The sub-queries are run in parallel, thus achieving gains in performance when compared to centralized approaches.

Virtual fragmentation is trickier in the XML model than in the relational model because of the lack of keys in the former. In fact, in the relational model, the sub-queries are built by adding selection predicates that range over the domain of the table key. The XPath function *position()* is a good replacement for the table key because it is unique (a given element in an XML document has a unique position in the context of its parent). However, this is not enough. In the same way that virtual fragmentation in the relational model needs the help of clustered indices, another feature of the Native XML databases is crucial for the virtual fragment to work properly - each element must be indexed by its position, so that the query processor can go directly and only access the desired elements when processing a sub-query. Full scans must be avoided at all costs. In fact, experimental results by Silva et al. [65] show that, in general, native DBMSX index XML elements by their position. Thus, virtual fragmentation is a promising technique, especially in dynamic environments such as clouds, but needs further investigation.

Parallelism. One of the main problems in obtaining acceleration in parallel query processing is load

balance, which is a problem in almost all scenarios. Load balance is very important for efficient query processing, and is not taken into account in existing approaches, not even in virtual fragmentation [62]. MapReduce tries to deal with load balancing by submitting backup tasks when the whole process is close to completion. This way, only the answer provided by the task that finishes first is considered [23]. This does not solve the problem, but it is a start. Recent work has been done on the issue of optimizing XML query processing in MapReduce [27, 28, 29]. Others leave the fragmentation technique open [19]. The authors of [62] suggest using adaptive virtual fragmentation [46] to solve this issue. This, however, has not been done yet.

Acknowledgements. We would like to thank Luiz Augusto Matos da Silva for helping in the bibliographic search. We would also like to thank CNPq and FAPERJ for partially supporting this research.

7. REFERENCES

[1] S. Abiteboul, A. Bonifati, G. Cobena, C. Cremarenco, F. Dragan, I. Manolescu, T. Milo, and N. Preda. Managing distributed workspaces with active XML. In *VLDB*, pages 1061–1064, 2003.

[2] S. Abiteboul, A. Bonifati, G. Cobena, I. Manolescu, and T. Milo. Dynamic XML documents with distribution and replication. In *SIGMOD*, pages 527–538, 2003.

[3] A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi, A. Silberschatz, and A. Rasin. HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads. *PVLDB*, 2(1):922–933, 2009.

[4] A. Andrade, G. Ruberg, F. Baião, V. Braganholo, and M. Mattoso. Efficiently processing XML queries over fragmented repositories with PartiX. In *DATAX*, pages 150–163, 2006.

[5] C. Baru, A. Gupta, B. Ludascher, R. Marciano, Y. Papakonstantinou, P. Velikhov, and V. Chu. XML-based information mediation with MIX. *SIGMOD Record*, 28(2):597–599, 1999.

[6] L. Birhanu, S. Atnafu, and F. Getahun. Native XML document fragmentation model. In *SITIS*, pages 233 –240, 2010.

[7] A. Bonifati and A. Cuzzocrea. Storing and retrieving XPath fragments in structured P2P networks. *DKE*, 59(2):247–269, 2006.

[8] A. Bonifati and A. Cuzzocrea. Efficient fragmentation of large XML documents. In *DEXA*, pages 539–550, 2007.

[9] A. Bonifati, U. Matrangolo, A. Cuzzocrea, and M. Jain. XPath lookup queries in P2P networks. In *WIDM*, pages 48–55, 2004.

[10] S. Bose and L. Fegaras. XFrag: a query processing framework for fragmented XML data. In *WebDB*, pages 97–102, 2005.

[11] S. Bose, L. Fegaras, D. Levine, and V. Chaluvadi. A query algebra for fragmented XML stream data. In *DBPL*, pages 195–215, 2003.

[12] J.-M. Bremer and M. Gertz. On distributing XML repositories. In *WebDB*, pages 73–78, 2003.

[13] J.-M. Bremer and M. Gertz. Integrating document and data retrieval based on XML. *The VLDB Journal*, 15(1):53–83, 2006.

[14] S. Chaudhuri, M. Datar, and V. Narasayya. Index selection for databases: a hardness study and a principled heuristic solution. *IEEE TKDE*, 16(11):1313 – 1323, 2004.

[15] D. Che, K. Aberer, and T. Ozsu. Query optimization in XML structured-document databases. *The VLDB Journal*, 15(3):263–289, 2006.

[16] D.-R. Che. Accomplishing deterministic XML query optimization. *Journal of Computer Science and Technology*, 20(3):357–366, 2005.

[17] H. Choi, K.-H. Lee, S.-H. Kim, Y.-J. Lee, and B. Moon. HadoopXML: a suite for parallel processing of massive XML data with multiple twig pattern queries. In *CIKM*, pages 2737–2739, 2012.

[18] C.-W. Chung, J.-K. Min, and K. Shim. APEX: an adaptive path index for XML data. In *SIGMOD*, pages 121–132, 2002.

[19] G. Cong, W. Fan, A. Kementsietsidis, J. Li, and X. Liu. Partial evaluation for distributed XPath query processing and beyond. *ACM TODS*, 37(4):32:1–32:43, 2012.

[20] B. F. Cooper, N. Sample, M. J. Franklin, G. R. Hjaltason, and M. Shadmon. A fast index for semistructured data. In *VLDB*, pages 341–350, 2001.

[21] D. Dash, N. Polyzotis, and A. Ailamaki. CoPhy: a scalable, portable, and interactive index advisor for large workloads. *PVLDB*, 4(6):362–372, 2011.

[22] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. In *OSDI*, pages 137–150, 2004.

[23] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. *CACM*, 51(1):107–113, 2008.

[24] J. Dean and S. Ghemawat. MapReduce: a flexible data processing tool. *CACM*, 53(1):72–77, 2010.

[25] J. Dittrich, J.-A. Quiané-Ruiz, A. Jindal, Y. Kargin, V. Setty, and J. Schad. Hadoop++: making a yellow elephant run like a cheetah (without it even noticing). *PVLDB*, 3(1-2):515–529, 2010.

[26] J. Dittrich, J.-A. Quiané-Ruiz, S. Richter, S. Schuh, A. Jindal, and J. Schad. Only aggressive elephants are fast elephants. *PVLDB*, 5(11):1591–1602, 2012.

[27] L. Fegaras. Supporting bulk synchronous parallelism in map-reduce queries. In *SC Companion: High Performance Computing, Networking Storage and Analysis*, pages 1068–1077, 2012.

[28] L. Fegaras, C. Li, and U. Gupta. An optimization framework for map-reduce queries. In *EDBT*, pages 26–37, 2012.

[29] L. Fegaras, C. Li, U. Gupta, and J. J. Philip. XML query optimization in map-reduce. In *WebDB*, pages 1–6, 2011.

[30] M. Fernandez, J. Simeon, and P. Wadler. An algebra for XML query. In *FST TCS*, pages 11–45, 2000.

[31] G. Figueiredo, V. Braganholo, and M. Mattoso. Processing queries over distributed XML databases. *JIDM*, 1(3):455–470, 2010.

[32] F. Frasincar, G.-J. Houben, and C. Pau. XAL: an algebra for XML query optimization. *Australasian Computer Science Communications*, 24(2):49–56, 2002.

[33] G. Gardarin, A. Mensch, T.-T. Dang-Ngoc, and L. Smit. Integrating heterogeneous data sources with XML and XQuery. In *DEXA*, pages 839–846, 2002.

[34] R. Goldman and J. Widom. DataGuides: enabling query formulation and optimization in semistructured databases. In *VLDB*, pages 436–445, 1997.

[35] G. Gou and R. Chirkova. Efficiently querying large XML data repositories: A survey. *IEEE TKDE*, 19(10):1381 –1403, 2007.

[36] H. Huo, G. Wang, X. Hui, R. Zhou, B. Ning, and C. Xiao. Efficient query processing for streamed XML fragments. In *Database Systems for Advanced Applications*, volume 3882 of *Lecture Notes in Computer Science*, pages 468–482. 2006.
[37] H. V. Jagadish, L. V. S. Lakshmanan, D. Srivastava, and K. Thompson. TAX: a tree algebra for XML. In *DBPL*, pages 149–164, 2001.
[38] C.-H. Jeong, Y. Choi, D.-S. Jin, M. Lee, S.-P. Choi, K. Kim, M.-H. Cho, W.-K. Joo, H.-M. Yoon, J.-H. Seo, and J. Kim. Service-centric object fragmentation for efficient retrieval and management of huge XML documents. In *PDCAT*, pages 118–124, 2007.
[39] K. Kido, T. Amagasa, and H. Kitagawa. Processing XPath queries in PC-Clusters using XML data partitioning. In *ICDE Workshops*, pages 114–119, 2006.
[40] J. Kim and H.-J. Kim. A partition index for XML and semi-structured data. *DKE*, 51(3):349–368, 2004.
[41] P. Kling, M. Ozsu, and K. Daudjee. Generating efficient execution plans for vertically partitioned XML databases. *PVLDB*, 4(1):1–11, 2010.
[42] P. Kling, M. Özsu, and K. Daudjee. Scaling XML query processing: distribution, localization and pruning. *Distributed and Parallel Databases*, 29(5):445–490, 2011.
[43] H. Kurita, K. Hatano, J. Miyazaki, and S. Uemura. Efficient query processing for large XML data in distributed environments. In *AINA*, pages 317–322, 2007.
[44] K. Lee, J. Min, and K. Park. A design and implementation of XML-Based mediation framework (XMF) for integration of internet information resources. In *HICSS*, pages 202–202, 2002.
[45] S. Lee, J. Kim, and H. Kang. Memory-efficient query processing over XML fragment stream with fragment labeling. *Computing and Informatics*, 29(5):757–782, 2010.
[46] A. Lima, M. Mattoso, and P. Valduriez. Adaptive virtual partitioning for OLAP query processing in a database cluster. *JIDM*, 1(1):75–88, 2010.
[47] H. Ma and K.-D. Schewe. Fragmentation of XML documents. In *SBBD*, pages 200–214, 2003.
[48] H. Ma and K.-D. Schewe. Heuristic horizontal XML fragmentation. In *CAISE*, pages 131–136, 2005.
[49] H. Ma and K.-D. Schewe. Fragmentation of XML documents. *JIDM*, 1(1):21–34, 2010.
[50] H. Ma and K.-D. Schewe. Revisiting "Fragmentation of XML documents". *JIDM*, 1(1):35–36, 2010.
[51] H. Ma, K.-D. Schewe, and Q. Wang. A heuristic approach to cost-efficient fragmentation and allocation of complex value databases. In *ADC*, pages 183–192, 2006.
[52] H. Ma, K.-D. Schewe, and Q. Wang. A heuristic approach to cost-efficient derived horizontal fragmentation of complex value databases. In *ADC*, pages 103–111, 2007.
[53] I. Machdi, T. Amagasa, and H. Kitagawa. XML data partitioning strategies to improve parallelism in parallel holistic twig joins. In *ICUIMC*, pages 471–480, 2009.
[54] M. Mattoso. Virtual partitioning. In L. Liu and M. T. Ozsu, editors, *Encyclopedia of Database Systems*, pages 3340–3341. 2009.
[55] J. McHugh and J. Widom. Query optimization for XML. In *VLDB*, pages 315–326, 1999.
[56] M. M. Moro, V. Braganholo, C. F. Dorneles, D. Duarte, R. Galante, and R. S. Mello. XML: some papers in a haystack. *SIGMOD Record*, 38(2):29–34, 2009.
[57] W. Ng and J. Cheng. An efficient index lattice for XML query evaluation. In *DASFAA*, pages 753–767, 2007.
[58] M. T. Ozsu and P. Valduriez. *Principles of Distributed Database Systems*. 3 edition, 2011.
[59] S. Paparizos, Y. Wu, L. V. S. Lakshmanan, and H. V. Jagadish. Tree logical classes for efficient evaluation of XQuery. In *SIGMO*, pages 71–82, 2004.
[60] Paul Grosso and Daniel Veillard. XML fragment interchange. W3C candidate recommendation 12 february 2001., 2001. W3C Candidate Recommendation 12 February 2001.
[61] A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, and M. Stonebraker. A comparison of approaches to large-scale data analysis. In *SIGMOD*, pages 165–178, 2009.
[62] C. Rodrigues, V. Braganholo, and M. Mattoso. Virtual partitioning ad-hoc queries over distributed XML databases. *JIDM*, 2(3):495–510, 2011.
[63] C. Sartiani and A. Albano. Yet another query algebra for XML data. In *Database Engineering and Applications Symposium*, pages 106–115, 2002.
[64] K.-D. Schewe. Fragmentation of object oriented and semistructured data. In *BalticDB*, pages 253–266, 2002.
[65] L. Silva, L. Silva, M. Mattoso, and V. Braganholo. On the performance of the position() XPath function. In *DocEng*, 2013.
[66] T. Silva, F. Baião, J. Sampaio, M. Mattoso, and V. Braganholo. Towards recommendations for horizontal XML fragmentation. *JIDM*, 4(1):27–36, 2013.
[67] M. Stonebraker, D. Abadi, D. J. DeWitt, S. Madden, E. Paulson, A. Pavlo, and A. Rasin. MapReduce and parallel DBMSs: friends or foes? *CACM*, 53:64–71, 2010.
[68] D. Suciu. Distributed query evaluation on semistructured data. *ACM TODS*, 27(1):1–62, 2002.
[69] B. Surjanto, N. Ritter, and H. Loeser. XML content management based on object-relational database technology. In *WISE*, pages 70–79, 2000.
[70] I. Tatarinov, E. Viglas, K. Beyer, J. Shanmugasundaram, and E. Shekita. Storing and querying ordered XML using a relational database system. In *SIGMOD*, 2002.
[71] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Antony, H. Liu, and R. Murthy. Hive - a petabyte scale data warehouse using hadoop. In *ICDE*, pages 996–1005, 2010.
[72] Z. Vagena, M. Moro, and V. Tsotras. Efficient processing of XML containment queries using partition-based schemes. In *IDEAS*, pages 161–170, 2004.
[73] M. Waldvogel, M. Kramis, and S. Graf. Distributing XML with focus on parallel evaluation. In *DBISP2P*, pages 55–67, 2008.
[74] Y. Wu, J. M. Patel, and H. V. Jagadish. Structural join order selection for XML query optimization. In *ICDE*, pages 443–454, 2003.
[75] B. B. Yao, M. T. Özsu, and J. Keenleyside. XBench - a family of benchmarks for XML DBMSs. In *Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web-Revised Papers*, pages 162–164, 2003.
[76] M. Zhang and J. T. Yao. XML algebras for data mining. In *Data Mining and Knowledge Discovery: theory, tools and technology*, pages 209–217, 2004.
[77] X. Zhang, B. Pielech, and E. A. Rundesnteiner. Honey, i shrunk the XQuery!: an XML algebra optimization approach. In *WIDM*, pages 15–22, 2002.

Approaches and Challenges in Database Intrusion Detection

Ricardo Jorge Santos
CISUC – DEI – FCTUC
University of Coimbra
3030-290 Coimbra – Portugal
lionsoftware.ricardo@gmail.com

Jorge Bernardino
CISUC – DEIS – ISEC
Polytechnic Institute of Coimbra
3030-190 Coimbra – Portugal
jorge@isec.pt

Marco Vieira
CISUC – DEI – FCTUC
University of Coimbra
3030-290 Coimbra – Portugal
mvieira@dei.uc.pt

ABSTRACT

Databases often support enterprise business and store its secrets. This means that securing them from data damage and information leakage is critical. In order to deal with intrusions against database systems, Database Intrusion Detection Systems (DIDS) are frequently used. This paper presents a survey on the main database intrusion detection techniques currently available and discusses the issues concerning their application at the database server layer. The identified weak spots show that most DIDS inadequately deal with many characteristics of specific database systems, such as *ad hoc* workloads and alert management issues in data warehousing environments, for example. Based on this analysis, research challenges are presented, and requirements and guidelines for the design of new or improved DIDS are proposed. The main finding is that the development and benchmarking of specifically tailored DIDS for the context in which they operate is a relevant issue, and remains a challenge. We trust this work provides a strong incentive to open the discussion between both the security and database research communities.

1. INTRODUCTION

Databases are of vital importance to nearly all enterprises. They support the business' operational and analytical requirements, and often store its secrets. For example, Data Warehouses (DWs) store extremely sensitive business information, making them a major target for attackers. Therefore, securing their data from damage or leakage is a critical issue. To manage this, enterprises typically implement several layers of protection between users and data, working at the network, host, and database levels.

Most solutions for data protection at the database layer consist of Database Intrusion Detection Systems (DIDS), well-defined data access policies and encryption. Although data access policies and standard encryption algorithms are widely used, and relatively simple to configure in today's DataBase Management Systems (DBMS), choosing which DIDS to use in certain environments is not a trivial task.

This paper describes and analyzes the main techniques for DIDS, and discusses their practical limitations. For example, since most database intrusion detection techniques rely on command-syntax analysis for user profiling (which typically consists of determining usual data access patterns and dependencies), the *ad hoc* nature of significant portions of data warehousing workloads makes distinguishing abnormal from normal user activity an extremely difficult task. Moreover, most DIDS are incapable of preventing or stopping a user action before it finishes its execution, *i.e.*, they lack intrusion response capability. Given the value of data in many business contexts, these are critical issues that might make the currently available DIDS inefficient or even unfeasible solutions in many database systems.

The main contributions of this work are the description of the distinct existing intrusion detection techniques and a discussion on how these DIDS are applicable to each database context, pointing out their weak spots considering the typical user workload and the specific characteristics of each type of environment. Based on those weaknesses, we present the existing research challenges and opportunities, and propose a set of requirements and guidelines to drive the development of new and/or improved DIDS specifically designed for those environments. We argue that this is a pertinent issue and remains a challenge.

Another important finding is that despite the importance of the role played by benchmarks in testing and comparing systems, until this moment no benchmark has been proposed for evaluating the performance of DIDS at the database level.

2. DATABASE INTRUSION DETECTION SYSTEMS

Detecting illicit access and malicious actions are the main goals of Intrusion Detection Systems (IDS). There are mainly two approaches: *misuse detection*, looking for well-known attack patterns; and *anomaly detection*, looking for deviations from typical user behavior. The first approach works efficiently against previously known and expected intrusion actions.

However, it is incapable of acting against intrusions that reveal new forms of attack or malicious user actions that seem "normal". To overcome these issues, anomaly detection techniques have been proposed.

In these systems there is typically a learning or training phase (*i.e.*, previous to intrusion detection), in which database logs and/or command datasets assumed as having "normal" or intrusion-free activity are used in order to build the user behavior profiles [21]. After this learning phase, the intrusion detectors match user actions against those profiles to find significant deviations which are signaled as potential intrusions.

The main requirements that intrusion detection systems need to cope with are:

1) Adequately defining and building profiles that accurately represent "normal" user behavior or workloads, as well as identifying attack signatures;
2) Given those profiles and/or attack signatures, define which behavioral features as well as which techniques and models maximize the performance and accuracy of the intrusion detection processes;
3) Reporting system status to security staff and notifying them about generated alerts;
4) Promote a way of stopping or preventing the attack whenever an intrusion alert is raised (this feature may or not be present in the IDS; if it is the case, literature often refers the IDS as an *Intrusion Detection and Response System*, or *Intrusion Detection and Prevention System*).

From an impersonation perspective, an intruder can be one of the following [30]:

- An *authorized user*, which is someone belonging to the enterprise that has regular access to authorized database interfaces and acts with malicious intent (commonly referred as an *insider threat*);
- A *masqueraded user*, which is someone that obtains the credentials of an authorized user and impersonating that user takes control of an authorized interface (also referred as an *insider threat* when the attacker is someone from within the enterprise but without regular authorized database access, and refers to an *outsider threat* when it comes from someone outside the enterprise that manages to obtain the credentials);
- An *external attacker* (commonly referred to as the *outsider threat*), which is someone from outside the enterprise that is able to bypass database security and gain direct database access using SQL injection or other vulnerability exploiting techniques.

Considering the intruders' intentions, there are mainly three types of attacks mobilized against databases [8]:

- *Attacks aiming at corrupting data (integrity attacks).* In these types of attack, the intruder seeks access to the database for executing actions that compromise its integrity, such as corrupting or deleting the data in a given database object (*e.g.* such as modifying the contents of a table);
- *Attacks aiming at stealing information (confidentiality attacks).* In these attacks, the intruder focuses on breaking confidentiality issues, such as stealing business information, rather than damaging data;
- *Attacks aiming at making the database unavailable (availability attacks).* These attacks aim on making database services unavailable to users, *i.e.*, they are mainly Denial of Service (DoS) attacks (*e.g.* flooding database services and bandwidth with a large number of requests, crashing database server instances, deleting database objects, etc).

In the past, several types of database intrusion detection techniques have been proposed. This section presents a descriptive analysis of selected samples from each different type of approach and/or technique for dealing with all types of attacks, in order to characterize the broad scope of existing solutions against both insider and outsider threats.

2.1 Temporal Analysis

These techniques focus on temporal features such as the time span between user actions and the duration of those actions. The approach in [18] uses a mean and standard deviation model built from time signatures to check for outliers within a predefined range in real-time database systems. This solution considers a transaction as a set of read and/or write actions for each data object which is executed in predefined update time periods.

For example, updating a temporal data object (event) can trigger a rule such that the update time is checked against the expected update time (condition) and rejects the update (action) if the predicate returns false, considering it an intrusion. The training period occurs until a significant mean with 99% confidence level of a normal distribution is obtained for each object/update pair. Database behavior is monitored by sensors at the transaction level, which are assumed to be small in size and have fixed semantics such as write-only operations and well-defined data access patterns. If a transaction tries to update a temporal data object that has already been updated in that period, an alarm is raised.

2.2 Dependency and Relation Analysis

Intrusion detection techniques based on dependency and relation analysis compute dependencies and/or relations among the distinct sets of user actions and/or accessed data to find out which columns, rows, tables, etc. and/or commands are usually issued or processed together.

The DEMIDS system [4] builds user profiles based on their activity by determining frequent itemsets from feature/value pairs and computes distance measures of user activity against the learnt frequent itemsets to detect intrusions, given a threshold. The features are typically based on the syntactical analysis of user commands, where the itemset domains are the sets of attributes issued together.

Another approach using frequent itemset mining is presented in [33]. This approach summarizes each user command into a tuple <Op, F, T, C> where Op is the type of SQL command (insert, select, etc), F is the set of attributes, T is the set of tables, C is the constrained condition set. An algorithm mines user query profiles using these tuples, based on the pattern of the submitted queries at the transaction level. The algorithm adapts the support and confidence of association rule mining by adding query structure and attribute relations to the computation.

The Role-Based Access Control DIDS proposed in [11] improves a previous approach [1] using features named quiplets for summarizing each user command. Considering a generic command `SELECT {Target-List} FROM {Relation-List} WHERE {Qualification}`, a quiplet is defined as (`C, PR, PA, SR, SA`) where `C` is the SQL main command (insert, select, etc.), `PR` is the Projection-Relation information, `PA` is the Projection-Attribute information, `SR` is the Selection-Relation information, and `SA` is the Selection-Attribute information.

The authors define three types of quiplets with different granularities: given a relation (alias table) `R1` with attributes `A1, B1, C1, D1` and a relation `R2` with attributes `A2, B2, C2, D2` and a user command `SELECT R1.A1,R1.C1,R2.B2, R2.D2 FROM R1,R2 WHERE R1.B1=R2.B2`, they generate the coarse c-quiplet (`select,<2>,<4>,<2>,<2>`), medium m-quiplet (`select,<1,1>,<2,2>,<1,1>,<1,1>`) and fine f-quiplet (`select,<1,1>,<[1,0,1,0],[0,1,0,1]>,<1,1>,<[0,1,0,0],[0,1,0,0]>`).

For anomaly detection when the database has role-based users (*i.e.*, it is possible to link each user action to a given role), a Naïve Bayes Classifier (NBC) is used as follows: for all queries in the audit logs, and for each role, the classifier for each type of quiplet is built (training phase); for each submitted query, if any of its classifiers is different from the ones in its roles, the action is considered an intrusion and an alert is generated (testing phase).

If role-based access policies are not implemented in the database, they propose unsupervised anomaly detection. In this case, positional and distance functions are defined for the quiplets and clustering techniques (*k*-centers and *k*-means) map every user to its representative cluster, which is the cluster with the highest number of training records for that user after the clustering phase (training phase). For each new query to test, two approaches can be used: 1) given the determination of its representative cluster, use the NBC as in the Role-Based anomaly detection to perform a similar test; or 2) verify if the new query is a statistical outlier using the MAD (Median of Absolute Deviations) test [24], which if true considers the action as an intrusion and generates an alert.

2.3 Sequence Alignment Analysis

Sequence alignment mainly consists in determining common sequences of events (such as commands, data attributes, accessed values, etc). DIDS using this type of techniques typically learn and identify the repeatable series of events with significant length and eventually break them into smaller-sized subsets to label or classify those sequences and their subsets as normal user behavior. In the detection phase, each sequence of new events is matched against the learnt user sequences and their subsets for measuring how they differ in order to evaluate its probability of being an intrusion.

The solution presented in [15] identifies sequences of accessed attributes, commands and tables for building user profiles. The proposed features are the command types (insert, select, update, etc.), attributes designated as sensitive, all attributes, operations on attributes, and mixes of all features. This work also defines criteria to choose among user-based, role-based or organization-based profiles, given the working context of the database. In the learning phase, it builds sequence models given a threshold for determining the maximum number of differences. In the detection phase, it also uses a threshold for computing the highest number of differences allowed between the tested sequences and those retained in the learning phase, to consider the sequences as normal or abnormal.

2.4 Integrating Dependency with Sequence Alignment Analysis

An approach for finding dependency relationships among transaction-level attributes with high support and confidence rules is proposed in [10]. They assume that whenever an attribute is updated, this action is

linked to a sequence of other events logged in the database (*e.g.* due to an update of a given attribute, other attributes are also read or written). Thus, each update is defined by three sets: the read set, a set of attributes that have been read because of the update; the pre-write set, a set of attributes that have been written before the update and because of it; and the post-write set, a set of attributes that have been written after the update as a consequence of it. Transactions that do not follow any of the mined data dependency rules are marked as malicious.

The work in [28, 29] improves that of [10] by considering attribute sensitivity, *i.e.*, giving a measure of importance to each attribute. It proposes three levels of attribute sensitivity, given its support in the analyzed transactions: high, medium and low. A weighted data mining algorithm is used to mine the dependencies between database attributes and generate rules that reflect that dependency, given the measured sequences of operations (read, write) and the sensitivity of each attribute. Any transaction not following these rules is identified as malicious. The authors also propose an extension to the Entity-Relationship (E-R) model to syntactically capture the sensitivity of the attributes.

A learning algorithm for representing transactions by directed graphs describing execution paths is proposed in [9]. New transaction sets that deviate from the learnt execution paths are seen as unauthorized sequences of SQL commands. The features used to build the execution paths are the command type (select, insert, delete, etc.), target objects (tables) and selected columns, and restriction attributes, all of which are obtained from typical DBMS audit entries [21] storing information on the UserID, SessionID, CommandID, TransactionID, user command, object owner, and a timestamp of its execution.

2.5 Statistical Analysis

Statistical analysis is used in several DIDS for computing user activity statistics. The approach in [27] makes use of statistical functions on reference values obtained from the data in relations (alias tables) and Δ-relations (changes of the values of the monitored objects/attributes for all reference values, per attribute, between two runs of the DIDS) for anomaly detection. An extension is defined as the set of all rows of an insertion/modification of data and a relation refers to a table or view. The reference values include count, minimum, maximum, average, standard deviation, ranges, computed ratios, zero length checking, and bit counting. A misuse detection method is also included, which works by examining database objects (Database, Function, Index, Privilege, Procedure, Rule, Schema, Statistics, Table, Trigger, and View) and all operations on them. This is done by previously defining if each pair <Database object, operation> is dangerous or not.

The work proposed in [19] is based on computing summarized statistics such as counting, maximum, minimum, mean, median, standard deviation and cardinality values of each attribute from the dataset resulting or affected by each user command. These statistics are stored in a vector with fixed dimension named as S-Vector, regardless of how large the command's result dataset may be. When the dataset for obtaining the S-Vector is large, the authors propose sampling the dataset by fetching the first initial *k* tuples or a subset of randomly picked *k* tuples to maintain performance and scalability. The set of each user's S-Vectors is then used for applying techniques such as clustering, naïve Bayes, support vector machines or decision trees in order to obtain models that represent the user's normal behavior given those S-Vectors. In the intrusion detection phase, statistical deviation and outlier verification is applied to inspect each user command and classify it as normal or abnormal.

2.6 Information-Theoretic Analysis

Approaches using information-theoretic analysis compute measures such as entropy and information gain for characterizing user profiles and compare them with those of subsequent actions to see how they differ from the original ones.

The work in [17] describes such an information-theoretic solution. Features are composed by a tuple of audit data with *n* variables for each data object (*e.g.* IP address, message size, etc). Entropy is used as a measure of regularity of audit data (*e.g.* event types such as a list of commands), where each record represents a class; the smaller the entropy, the fewer the number of distinct records (*i.e.*, there is a higher number of redundancies), indicating more regular audit datasets. The fact that many events are repeated (or redundant) in a dataset suggests that they are likely to appear in the future. Anomaly detection models built from datasets with small entropy will likely be simpler and have better detection performance.

Conditional entropy is used to define temporal sequences of audit data. $H(X|Y)$ shows how much uncertainty remains for the remaining audit events in a sequence X after seeing Y. For anomaly detection, it is used as a measure of regularity of sequential dependencies. If the audit trail is a sequence of events of the same type, then the conditional entropy is 0 and the event sequences are deterministic. Conversely, large conditional entropy indicates that the sequences are not as deterministic and hence much harder to model.

Relative conditional entropy between distributions is used to measure regularities (distance) between two audit datasets, where the training dataset is a validated audit dataset and the tested dataset is the one to be inspected. Once again, the best solution is the one with smaller relative conditional entropy. Information gain is introduced to aid the feature selection and construction process to improve the detection performance because of its direct connection with conditional entropy. The higher information gain owned by the feature, the smaller conditional entropy, and hence the better detection performance.

2.7 Command Template Analysis

Command modeling DIDS use a command log to analyze all regular user commands and build summarized templates that generically represent the typical user workloads.

In [16], an algorithm summarizes a set of supposed "legitimate" queries into SQL templates that represent the models of all the queries. Each conditional filtering variables in the WHERE clause of similar commands are considered as parameters. To see if an unbounded variable or a finite list of values should be used for each parameter, a Kolmogorov-Smirnov test is done at a 90% confidence level. The algorithm also tabulates the frequency of each learnt fingerprint, *i.e.*, how often it occurs in the set of SQL statements.

Taking a new fingerprint `F` and a previously defined fingerprint `F′`, `F` is considered legitimate if `F` differs from `F′` only by: 1) any extra conditions in the WHERE clause of `F` that are missing from `F′` are joined with the AND operator; and 2) `F` selects an equal or fewer number of columns than `F′`. This work also proposes a method to deduce missing fingerprints (*i.e.*, ranges of queries similar to the database log queries used in the learning phase), based on mixing the possible combination of conditions in the WHERE clause from the previously acquired fingerprints. In the testing phase, each command significantly differing from the computed fingerprints is considered abnormal.

In [2] the authors propose applying a grammar-based analysis using tree-kernel based machine-learning techniques instead of commonly used vector-based data. This approach uses the parse-tree structure of SQL for correlating commands with applications and to differentiate between benign and malicious ones by inspecting changes in command syntax trees. They derive a distance measure induced by a tree-kernel function to measure the similarity of SQL commands using their parse-trees. Support vector machines are used in the learning phase and clustering is applied for distinguishing benign from malicious commands by outlier detection. This method promotes a context sensitive similarity that enables locating the nearest non-intrusive command for a malicious statement, which helps in root cause analysis.

3. INTRUSION RESPONSE AND PREVENTION

In what concerns intrusion response and prevention, which is the capability of stopping the intrusion action when it occurs or even before it occurs, it can be seen that several solutions enable full intrusion prevention, while others can only partially accomplish this.

In [18], the temporal analysis technique detects any queries that request execution outside a predefined time schedule and may therefore deny their execution and prevent the intrusion action. The sequence analysis technique used in [15] may enable intrusion prevention by avoiding subsequent user actions when it detects a suspicious sequence of actions. However, it needs to wait for a significant amount of actions that make up that sequence, meaning that it will probably only detect the intrusion after some of those actions have finished their execution, which makes it only capable of partial intrusion prevention.

All the solutions based on dependency and relational analysis that were described [1, 4, 11, 33] are fully capable of enabling intrusion prevention, since they may check each individual user command syntax and if they find those commands suspicious their execution can be stopped before their execution occurs. The solutions integrating a mix of dependency and sequence analysis such as [9, 10, 28, 29] are capable of performing only partial intrusion prevention, for the same reasons pointed out in the previous paragraph concerning the solution proposed in [15].

The work in [12] proposes a DIDS with intrusion detection and response mechanisms, improving a previous proposal in [11]. They propose defining database response policies and deal with potential intrusions using policy matching. The authors propose set of SQL-like rules in a syntax as *ON {Event} IF {Condition} THEN {Action} CONFIRM {Confirmation Action} ON SUCCESS {Resolution Action} ON FAILURE {Failure Action}* that will enable security staff to define those policies and determine what sort of actions the DIDS should take against intrusions.

The anomaly attributes used as intrusion detection features are the UserId, his/her role, client application, source IP address, and date/time of each user action, and the database, schema, object type (table, view, etc), the SQL command and its attributes. An administration model is included to manage the response policies and present algorithms for efficiently searching the policies

matching an anomalous user request in the policy database. The possible responses to intrusion actions can be {Do nothing, Log anomaly details, Send notification, Taint or Suspend user action, Abort or Disconnect user, Revoke or Deny user privileges}.

The solutions presented in [19, 27], based on statistical analysis, are mostly incapable of intrusion prevention, as they mostly rely on analyzing the changes in data or execution results *after* they have been processed. This means they can only detect the intrusion *a posteriori* to the attack. However, the approach in [27] can be adapted to check *a priori* statistical data concerning the rows requested to be processed, enabling partial intrusion prevention capabilities. For this same reason, the information-theory analysis approach presented in [17] may also accomplish partial intrusion prevention.

The solutions based on command and template analysis in [2, 16] can fully enable intrusion prevention due to same reason as those previously mentioned for dependency and relational analysis [1, 4, 11, 33].

Besides the previously described specific intrusion detection techniques and approaches that can be used in databases, other research works have been published that can also contribute to this intrusion detection field. For example, although it does not present itself as a DIDS, the work in [20] describes a method for auditing SQL queries to measure their suspiciousness from a privacy and confidentiality perspective that may be useful for intrusion detection purposes. A generic survey on how data mining techniques can be applied to intrusion detection is shown in [23], and an extensive survey on SQL injection is given in [14].

Table 1 summarizes the techniques described in this section, referring each type of technique along with the actions and user action elements that can be analyzed. It also shows if each approach allows implementing intrusion prevention, *i.e.*, if it enables stopping the intrusion action *a priori* to its execution.

4. APPLICATION OF INTRUSION DETECTION IN DATABASES

The applicability of DIDS in database systems depends on the type of environment in which they are supposed to operate. Understanding the characteristics inherent to the typical workloads of each type of environment is critical to determine which type or class of Intrusion Detection (ID) techniques can be more efficient given the nature of those workloads and thus, be considered as more adequate for the specific database system.

4.1 Transactional versus Analytical Database Systems

In an enterprise, the transactional (alias operational) systems typically consist of a set of applications and data sources that enable accomplishing and storing business transactions, and guarantee their operability [13]. Transactional databases are designed to manage the data required in supporting individual business transaction instead of cross-enterprise business analysis. Transactional systems typically consist of many users reading and writing small amounts of data. For example, an ATM bank system can have hundreds or thousands of users accessing their account balances at the same time or withdrawing/transferring a given amount of money. Another characteristic of the system is that it does not require keeping long periods of historical data; it only needs the current balance and latest movement records to be able to adequately support user requests and business transactions.

Table 1. Database intrusion detection techniques and their coverage

Technique	Reference	Elements that can be analyzed				Intrusion Prevention Capability
		Command Syntax	Accessed Columns	Processed Rows	Result Dataset	
Temporal Analysis	Lee, 2000 [18]	X				Yes
Dependency and Relation Analysis	Chung, 1999 [4]	X	X			Yes
	Zhong, 2004 [33]	X	X	X		Yes
	Bertino, 2005 [1]	X	X			Yes
	Kamra, 2008 and 2010 [11, 12]	X	X			Yes
Sequence Alignment Analysis	Kundu, 2010 [15]	X				Partial
Integrated Dependency with Sequence Alignment Analysis	Hu, 2004 [10]	X	X			Partial
	Srivastava, 2006 [28, 29]	X	X			Partial
	Fonseca, 2008 [9]	X	X			Partial
Statistical Analysis	Spalka, 2005 [27]	X	X	X		Partial
	Mathew, 2010 [19]	X	X		X	No
Information-Theoretic Analysis	Lee, 2001 [17]	X				Partial
Command Template Analysis	Lee, 2002 [16]	X	X			Yes
	Bockermann, 2009 [2]	X	X			Yes

In contrast, analytical systems are usually accessed by fewer users that query large amounts of data to obtain business analysis information to aid decision making. Using the same bank ATM system as an example, the difference is that the people from the bank that need to make decisions regarding the business (*i.e.*, managers, administrators, etc.) want to know the average balance for the last six months or a year for the accounts within a certain geographical region, for instance, in order to aid strategic decisions like opening a new branch office or encourage people to increase their investments by offering better interest rates. To execute this kind of query, the system needs to keep historical data of the balances plus it would read millions of records of all clients within a certain region to compute that average.

This type of analytical actions result in very demanding data access patterns, that if running on top of a transactional database can lock large amounts of data and consume computational resources in a way that could compromise the transactional system's availability. Ultimately, this could make it incapable of supporting the business transactions.

To relieve resource consumption, reduce operational risk in the transactional applications that support business and provide an optimized data structure for analytical cross-enterprise decision support purposes, Data Warehouses (DWs) are used. DWs clearly separate the analytical business processes from the transactional business processes. According to [13], we can assume the following distinct characteristics between transactional and analytical systems:

- From a perspective attending to its purpose, a DW is mainly a database system specifically designed for providing decision support information and business knowledge, while an operational system is specifically designed to support individual business transactions. Given that the business often requires the operational system to be online in order to accomplish a transaction, operational system requirements focus on enabling high availability to avoid risk in the accomplishment of the transactions themselves. On the other hand, since most decision support queries often require processing a large amount of data, DWs focus on fast query performance with high data throughput [13].
- From a perspective attending to the size and shape of its contents, a DW is composed of consolidated historical business data, mostly conformed within data schemas that optimize the execution of analytical queries. Generally, storing the business history implies taking up a very large amount of storage space, which often ranges from gigabytes to terabytes. In contrast, operational systems aim to keep their data sources "light", *i.e.*, small in size to minimize processing efforts and consequently keep their availability as high as possible. Transactional systems therefore keep only the exact amount of data which is required to support current and near-future business transactions.
- In what concerns their data schemas, transactional databases mostly have highly normalized schemas with a large number of tables and relationships amongst them, mainly to avoid data redundancy and keep each table small-sized, while DWs have denormalized schemas. Most DW database schemas are based on star schemas, where business facts are stored in a central table called fact table (*e.g.* sales table) and tables containing the business descriptors are called dimension tables (*e.g.* customer and product tables) [13]. Dimension tables link to the fact table by their primary keys (*e.g.* CustomerID, ProductID), are usually small in size (typically less than 10% of total storage space) and have a small amount of rows (up to tens of thousands), when compared with fact tables, which are typically very large in size and a huge amount of rows (millions or billions). Business facts are mainly stored in numerical-typed attributes within fact tables; since fact tables typically take up at least 90% of the DW total storage size [13].
- Considering his/her responsibility in the business, the DW user is typically a business manager or someone having a role of responsibility in the enterprise, while the typical user of operational systems are mainly transactional operators with low responsibility and few or none decision making privileges. Since they mainly consist of business managers and decision makers, the number of DW users it typically low (a few tens), while in many transactional systems the number of users is relatively high (tens to thousands).
- While end users of operational systems typically execute intensive read and write instructions, DW end users only execute read-only instructions, while DBAs and ETL (Extract-Transform-Load) users may insert or modify data. More than 90% of actions in DWs are typically analytical queries (*i.e.* SELECT statements), mainly executed against fact tables [13]. Reporting (*i.e.* periodically running reports for answering predefined decision support queries) is typical in DWs. Besides reporting, in many cases a very significant amount of decision support queries are *ad hoc*, which makes them mostly unpredictable in their syntax and frequency. In operational systems the queries are mainly simple, predefined and repetitive.

- Although analytical queries may typically access huge amounts of data, their response usually results in small datasets with a few hundred bytes and a relatively low number of columns (no more than a few tens). Most queries in DWs are CPU intensive and can take up to hours, while operational system queries are intended to be computationally fast and deliver very small response time.

Table 2 summarizes the main differences between operational systems and DWs, based on [13]. We shall now discuss how each type of ID technique is able to handle the characteristics inherent to each distinct type of database system.

4.2 Applying Intrusion Detection to Database Systems

As shown in Table 1, most DIDS focus on analyzing user command syntax (*i.e.*, parsing the SQL-expression syntax of queries to construct user profiles). As pointed out in [19], the most common problems with this type of approach are:

- Regular user queries may differ widely in syntax yet produce "normal" (*i.e.*, good non-intrusive) output, which generates false positives (*i.e.*, false alarms);
- Queries may be crafted by the attacker to differ slightly in syntax from the "normal" user behavior profiles yet produce "abnormal" (*i.e.*, malicious and intrusive) output, which generates false negatives (*i.e.*, attacks that pass undetected).

Given the expressiveness of the SQL language and the need to determine query equivalence or similarity, syntax analysis is complex and very difficult to perform correctly. In fact, query containment and equivalence is NP-complete for conjunctive queries and uncertain for queries involving negation [19].

In databases where typical user workloads have a well-defined number of distinct commands that are issued repetitively, relying on command syntax analysis may be feasible to achieve high ID efficiency. This is typically what occurs in transactional systems. However, in analytical systems such as DW's many actions are *ad hoc* and have variable execution times with variable data access patterns and dimension-size frequencies and thus, are mostly unpredictable and broad-scoped. This makes distinguishing between normal and abnormal commands in DWs an extremely difficult task. In such analytical databases, limiting ID to command syntax analysis by simply modeling SQL command templates or static frequent data access patterns (*e.g.* which tables or columns are accessed) is unreliable or, at least, minimalist.

Regarding the characteristics of DW user workloads, the ID solutions relying on temporal analysis such as presented in [18] are inadequate and mostly produce very poor ID results due to the unpredictable rate and execution time of those workloads. Due to the *ad hoc* nature of most of those workloads, ID solutions such as [2, 16] that are based on command template analysis lack the necessary dynamics to efficiently perform the ID processes and therefore also produce poor ID results. In transactional systems, temporal analysis is very efficient when the user actions occur within well-defined time periods and have predictable processing times. Otherwise, it suffers from the same issues with temporal analysis as those in DWs.

Table 2. Differences between Operational Systems and Data Warehouses

	Operational Systems	**Data Warehouses**
Workload nature/purpose	Transactional	Analytical
Temporal nature of the data	Current	Historical and current
Typical database storage size	As small as possible	Very large to huge
Typical number of tables	Medium to high	Small
Typical data schema type	Highly normalized	Denormalized
Typical number of users	Medium to large	Small
Typical user's business responsibility	Low	High
Typical type of command	Read/Write of small amounts of data	Read-only on large amounts of data
Typical command complexity	Simple	Medium to High
Typical operation dynamics	Static, predefined, predictable, repetitive	Reporting + Dynamic, *ad hoc*, iterative
Typical command response time	Small	Large
Typical command action	Read/write of a single row or few rows	Reporting and aggregation on many rows
Amount of data typically processed by each command	Small	Large or Very Large
Typical data update frequency	Often in a given period of time	Once periodically
Dataset size typically resulting from a command execution	Small	Variable (often Small)

Although the approach proposed in [19] adds a data-centric analysis of each user command execution's resulting dataset, the analysis is *a posteriori* to that execution. Given the time span between the start of the intrusion and its detection, together with resource consumption and sensitivity of the targeted data, many enterprises can suffer huge losses in case the intrusions compromise the system's availability or leak out business secrets, if their DIDS either takes too long to alert a malicious intrusion or is unable to prevent or stop its execution. In this sense, approaches *a posteriori* are not efficient solutions for ID in both transactional databases as well as DWs.

Conclusively, the unpredictable execution frequency and *ad hoc* nature of DW user workloads make time-based and SQL modeling ID approaches such as [2, 16, 18] mostly inadequate. Alternatively, DIDS performing ID at a coarse-grained basis such as database sessions or transaction command sets, instead of a fine-grained basis such as analyzing each SQL command, risk that a series of malicious commands may be executed before the intrusion can be dealt with. Therefore, data dependency and sequence alignment approaches such as [4, 33] that are able to inspect each user command *a priori* to its execution but only after a considerable amount of actions have been executed, should be carefully used according to each database context.

Data-centric techniques such as [19, 27] are capable of adding value to *a priori* ID techniques by executing an *a posteriori* analysis of the data affected by the user action. Combining these techniques with data access pattern analysis techniques such as [1, 11, 12], that deem the processed data, seem the most feasible and efficient DIDS for both types of database systems.

5. RESEARCH CHALLENGES

Considering the discussed issues, this section points out research challenges and guidelines for evaluating, developing and improving DIDS.

5.1 Intrusion Activity and Data Coverage

Command syntax-centric approaches focus on attack *syntax*, while data-centric approaches focus on its *semantics*. Distinguishing attack queries that have resulting datasets whose columns and resulting rows significantly differ from those of normal queries is covered by both syntax-centric and data-centric approaches, while data-centric approaches mostly capture attack queries that have similar columns but process or display different row contents from those of normal queries. Attack queries that are similar in both columns and resulting datasets are more easily discovered by syntax-centric approaches than by data-centric approaches.

To determine user intent, DIDS should not only focus on *how*, but also *what* data is *accessed* and *processed* (*i.e.*, which tables and columns, as well as which rows, are involved in the command's execution), and also *generated as a result* (*i.e.*, the resulting dataset's rows and columns). Besides the user command, the accessed and processed data and resulting datasets should be object of analysis. As far as we know, there is no DIDS approach mixing these items and covering this type of integrated broad-scope analysis.

5.2 Alert Management

Regardless of the ID technique, thresholds are typically used to define the probability of a certain action being an intrusion or compute if a particular alert should be considered significant or not. The main issue in these procedures is that high thresholds may allow many intrusions to pass undetected, while low thresholds may generate huge amounts of alerts, most of which probably refer to false alarms (alias false positives).

Given the sensitivity of data in many database systems, it is preferable to have low thresholds because the potential cost of non-detection is often too high or unacceptable. However, even slight changes to the parameters used in data mining and statistical DIDS may result in a huge, even exponential, increase of generated alerts. In this case, the number of false alarms is often so large that it leads to wasting immense time and resources, or the amount of alerts is so high that they are not possible to check in practice [25, 26, 28, 29]. This lowers the DIDS' efficiency and jeopardizes its feasibility, usefulness, and credibility.

Alert correlation techniques such as [6, 22, 25, 26, 31] have been proposed to deal with large amounts of generated alerts and decrease false positive rates, by grouping sets of alerts in order to apply some sort of classification that allows them to conclude which alerts are most probable of referring a true alert. Although they effectively reduce the number of alerts to check as well as the number of false alarms, we argue that they are not the best choice for alert management.

Since these techniques rely on filtering alerts, they may allow critical true intrusions – capable of producing a high amount of damage - to pass undetected, although they were initially flagged. We argue that no alert should be discarded and all alerts should be considered using alert ranking techniques instead, assessing the impact to the enterprise that might be caused by the intrusion to which the alert refers. Ranking the alerts improves damage or leakage containment by pointing out the intrusions that might cause more damage to the enterprise, so they can be rapidly dealt with.

5.3 Intrusion Impact Evaluation

To the best of our knowledge, there is no DIDS that evaluates the potential impact (*i.e.*, damage) that each potential intrusion is capable of doing to the database and/or enterprise. Given the business value of many database systems (*e.g.* data warehouses), this is a decisive issue to enable quickly dealing with threats representing high risk to the enterprise. Although approaches such as [28, 29] consider a measure of sensitivity for each attribute, no DIDS enables the assessment of the data itself that can be damaged or affected by the intrusion.

Besides detecting intrusions, DIDS should be able to measure or estimate a measure of the *impact* that could be produced by the intrusion. The main challenge is to determine *how sensitive* is the data targeted by the attack. Having the capability of measuring the impact of a given intrusion would allow classifying each intrusion action as tolerable or critical to the enterprise. As we previously mentioned, this could play a very important role in alert management in environments where large amounts of alerts are generated. Given that most alerts in these environments refer to false alarms, focusing on those that are in fact the most important ones would potentiate an efficient administration of intrusion response actions and resources.

5.4 Real-time Intrusion Detection, Response and Prevention Capabilities

Many DIDS execute the ID process *a posteriori*, *i.e.*, after the intrusion action has finished its execution, or are able to detect the intrusion while it occurs but are not able to stop it. Once again, given the value of data in many database systems and the potential costs of damage or information leakage, we consider the capability of a DIDS to detect and respond to intrusions in real-time as a critical requirement, *i.e.*, it must be capable of responding to an intrusion while it occurs and preferably before it produces any damage.

5.5 Evolution of ID Efficiency

DIDS should be able to automatically tune their ID algorithms in order to improve their efficiency by learning from their false positive and false negative results. They should enable calibrating their features, statistical functions and tests, classifiers and any other element belonging to the DIDS, and propose a method as how to achieve this. Such an approach for network ID has been proposed in [32], but as far as we know no similar solution has been proposed for automatically tuning DIDS. Machine learning techniques or other techniques that enable incrementally adjusting their ID parameters for improving their efficiency are advisable.

Another approach to improve DIDS is focusing on the typical characteristics of the database system in which they operate, *i.e.*, separating DIDS meant for operational systems from those meant for analytical systems. Given the distinct workloads between transactional and analytical databases, specifically tailored ID techniques for highly transactional environments as opposed to ID techniques for DWs should be able to achieve higher efficiency than those referred to as "all-in-one" general solutions.

5.6 Database Intrusion Detection Benchmarking

We acknowledge the fact that an experimental evaluation of a database intrusion workload setup using the ID techniques described in this paper would bring added value to this work, as well as support its discussion and conclusions. However, the datasets and attack loads used in database ID research are mostly synthetic and several came from proprietary real-world datasets, which makes them unusable for third parties. In fact, the only benchmark commonly used by several solutions was the KDD99 [5]; all the remaining used synthetically generated workloads and datasets or specific datasets from real-world scenarios.

This is the main reason why we do not discuss ID efficiency results from these publications. Furthermore, although the use of advanced techniques such as Support Vector Machines and artificial neural networks might suggest obtaining better results than simpler ones such as statistical measures, this is not clear or demonstrable at this point for the same reasons. Therefore, benchmarks are an essential instrument in the development and implementation of many systems. They are widely used for two main reasons:

- Benchmarks provide a mean to test those systems and supply solution providers and clients with measures that enables a meaningful comparison between different alternatives;
- They also provide relevant feedback to developers which enables them to improve those solutions.

Since the KDD99 benchmark focuses on intrusions at the network and operating system (OS) level, in what concerns databases a need arises for dealing with intruders that are able to bypass ID mechanisms working at the network and OS level. In spite of the criticality of protecting data against intrusions and the importance of having available benchmarks for testing and improving DIDS, there is no benchmark focusing on workloads at the database command level.

6. CONCLUSIONS

We have presented a survey on the available ID techniques and approaches used in DIDS and pointed out the issues that concern their usage. We argue that distinct database systems have unique user and data processing requirements that differ from each other and require distinctively tailored ID approaches. The difficulty in accurately profiling user behavior, the overstated number of alerts and false alarms generated by most ID techniques, the potentially low reliability on correlation techniques and the hypothesis that many intrusions may only be detected and dealt with *a posteriori* to the attacks and without any knowledge on the type of damage that the intrusions might produce, jeopardize the credibility and feasibility of DIDS in many specific high sensitive data contexts such as data warehousing environments [2, 25, 26].

Considering the typical very specific user workloads of distinct types of database systems and the importance of databases for enterprises, we conclude that specific DIDS should be developed, pursuing the following requirements and guidelines:

- Although role or session profiling can be used, a DIDS should be more accurate and efficient if it is able to manage individual user profiles;
- All user actions must be traceable, *i.e.*, the DIDS must be able to trace the user and IP address it comes from and the session to which it belongs;
- Perform real-time intrusion action analysis and have near real-time intrusion prevention and response mechanisms, as in [7, 12], preferably before each user command is executed;
- User action analysis for building profiles and executing ID must focus on fourfold items: user commands, accessed data, processed data, and execution results;
- Impact evaluation, *i.e.*, measuring the damage to the enterprise that might occur as a result of the intrusion action should be used for alert management for optimizing intrusion response;
- The ID techniques should be able to evolve through time, *i.e.*, they should be able to learn from each confirmed intrusion alert or false alarm and tune algorithms or models to increase detection rates and decrease false alarm rates;
- The availability and correct operation of the DIDS and the database must be mutually verifiable;
- When the DIDS is unable to prevent the execution of intrusive actions, it should consider the execution of a recovery-from-attack type solution, such as [3];
- Given the database performance issues in very large database systems such as DWs, the DIDS security aptitudes must seamlessly operate in settings with strict performance and scalability requirements.

To the best of our knowledge, no DIDS has been proposed that has been proved capable of efficiently complying with all the referred requirements and guidelines proposed in this paper.

Additionally, a DIDS benchmark for each type of database system should be developed and proposed by both the research community and industry. We acknowledge that defining benchmarks is not a trivial task and that there are always discussable issues concerning the objectivity and effectiveness of each proposal. A DIDS benchmark should provide a wide coverage of possible intrusion activity according to the several distinct user workloads, while simulating their execution in a realistic-like environment. Given the importance of ID in specific contexts such as DWs and the lack of standard benchmarks for testing DIDS at the SQL level, we believe that the issues presented in this paper are worthy of notice and hope that our work may motivate the discussion around the subject in both database and intrusion detection research communities, possibly driving the development of a standard benchmark for this purpose.

7. ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers of the ACM SIGMOD Record for their helpful comments. This paper was partially supported by project iCIS – Intelligent Computing in the Internet Services (CENTRO-07-ST24 – FEDER – 002003), Portugal.

8. REFERENCES

[1] Bertino, E., Kamra, A., Terzi, E. and A. Vakali. "Intrusion Detection in RBAC-Administered Databases", Annual Computer Security Applications Conference (ACSAC), 2005.

[2] Bockermann, C., Apel, M. and M. Meier, "Learning SQL for Database Intrusion Detection using Context-Sensitive Modeling", International Conference on Knowledge Discovery and Machine Learning (KDML), 2009.

[3] Chakraborty, A., Majumdar, A. K. and S. Sural, "A Column Dependency-Based Approach for Static and Dynamic Recovery of Databases from Malicious Transactions", International Journal of Information Security (9), 2010.

[4] Chung, C. Y., Gertz, M. and K. Levitt, "DEMIDS: A Misuse Detection System for Database Systems", IFIP TC11 WG11.5 Conf. on Integrity and Internal Control in Information Systems, Kluwer Academic Publishers, 1999.

[5] DARPA archive, Task Description of the KDD99 Benchmark, available at http://www.kdd.ics.uci.edu/databases/kddcup99/task.html.
[6] Debar, H., and A. Wespi, "Aggregation and Correlation of Intrusion-Detection Alerts", Recent Advances in Intrusion Detection (RAID), 2001.
[7] Dia, J., and H. Miao, "D_DIPS: An Intrusion Prevention System for Database Security", Int. Conference on Information and Communications Security (ICICS), 2005.
[8] Douligeris, C. and A. Mitrokotsa, "DDoS Attacks and Defense Mechanisms: Classification and State-of-the-Art", Int. Journal of Computer Networks (IJCN), Elsevier B. V., 44, 2004.
[9] Fonseca, J., Vieira, M. and H. Madeira, "Online Detection of Malicious Data Access using DBMS Auditing". ACM Int. Symposium on Applied Computing (SAC), 2008.
[10] Hu, Y. and B. Panda, "A Data Mining Approach for Database Intrusion Detection". ACM Intern. Symposium on Applied Computing (SAC), 2004.
[11] Kamra, A., Terzi, E. and E. Bertino, "Detecting Anomalous Access Patterns in Relational Databases". Springer VLDB Journal, 17, 2008.
[12] Kamra, A. and E. Bertino, "Design and Implementation of an Intrusion Response System for Relational Databases", IEEE Transactions on Knowledge and Data Engineering (TKDE), Vol. 23, No. 6, June 2011.
[13] Kimball, R. and M. Ross, The Data Warehouse Toolkit, 3rd Ed. Wiley & Sons, Inc., 2013.
[14] Kindy, D. A. and A. K. Pathan, "A Detailed Survey on Various Aspects of SQL Injection: Vulnerabilities, Innovative Attacks and Remedies", Int. Journal of Communication Networks and Information Security (IJCNIS), Vol. 5, No. 2, August 2013.
[15] Kundu, A., Sural, S. and A. K. Majumdar, "Database Intrusion Detection Using Sequence Alignment". International Journal of Information Security (9), 2010.
[16] Lee, S. Y., Low, W. L. and P. Y. Wong, "Learning Fingerprints for a Database Intrusion Detection System". Euro Symposium on Research in Computer Security (ESORICS), 2002.
[17] Lee, W. and D. Xiang, "Information-Theoretic Measures for Anomaly Detection", IEEE Symposium on Security and Privacy, 2001.
[18] Lee, V. C. S., Stankovic, J. A. and S. H. Son, "Intrusion Detection in Real-time Database Systems via Time Signatures". Real-time Technology and App. Symposium (RTAS), 2000.
[19] Mathew, S., Petropoulos, M., Ngo, H. Q. and S. Upadhyaya, "A Data-Centric Approach to Insider Attack Detection in Database Systems". International Conference on Recent Advances in Intrusion Detection (RAID), 2010.
[20] Motwani, R., Nabar, S. U. and D. Thomas, "Auditing SQL Queries", Int. Conf. on Data Engineering (ICDE), 2008.
[21] Newman, A. C., "Intrusion Detection and Security Auditing in Oracle". Application Security Inc. White Paper, 2011.
[22] Ning, P., Cui, Y. and D. S. Reeves, "Analyzing Intensive Intrusion Alerts via Correlation", Recent Advances in Int. Detection (RAID), 2002.
[23] Pei, J., Upadhyaya, S. J., Farooq, F. and V. Govindaraju, "Data Mining for Intrusion Detection: Techniques, Applications and Systems", Int. Conf. on Data Engineering (ICDE), 2004.
[24] Pham-Gia, T. and T. L. Hung, "The Mean and Median Absolute Deviations", International Journal on Mathematical and Computer Modelling", Vol. 34, Issues 7-8, October 2001.
[25] Pietraszek, T., "Using Adaptive Alert Classification to Reduce False Positives in Intrusion Detection". Int. Conf. on Recent Advances in Intrusion Detection (RAID), 2004.
[26] Pietraszek, T. and A. Tanner, "Data Mining and Machine Learning – Towards Reducing False Positives in Intrusion Detection". Inf. Security Technical Report, 10(3), 2005.
[27] Spalka, A. and J. Lehnhardt, "A Comprehensive Approach to Anomaly Detection in Relational Databases". IFIP Int. Conf. Data and Applications Security and Privacy (DBSec), 2005.
[28] Srivastava, A., Sural, S. and A. K. Majumdar, "Database Intrusion Detection using Weighted Sequence Mining". Journal of Computers, Vol. I, No. 4, 2006.
[29] Srivastava, A., Sural, S. and A. K. Majumdar, "Weighted Intra-Transactional Rule Mining for Database Intrusion Detection". Int. Pacific-Asia Conference on Knowledge Discovery in Databases (PAKDD), 2006.
[30] Treinen, J. and R. Thurimella, "A Framework for the Application of Association Rule Mining in Large Intrusion Detection Infrastructures", International Conference on Recent Advances in Intrusion Detection (RAID), 2006.
[31] Valdes, A. and K. Skinner, "Probabilistic Alert Correlation". International Conference on Recent Advances in Intrusion Detection (RAID), 2001.
[32] Yu, Z., Tsai, J. P. and T. Weigert, "An Automatically Tuning Intrusion Detection System". IEEE Transactions on Systems, Man, and Cybernetics, Vol. 37, No. 2, 2007.
[33] Zhong, Y. and X. Qin, "Database Intrusion Detection Based on User Query Frequent Itemsets Mining with Item Constraints", Information Security Conf. (InfoSecu), 2004.

Text-Mining, Structured Queries, and Knowledge Management on Web Document Corpora

Hamid Mousavi
CSD, UCLA
Los Angeles, CA
hmousavi@cs.ucla.edu

Maurizio Atzori
University of Cagliari
Cagliari, Italy
atzori@unica.it

Shi Gao
CSD, UCLA
Los Angeles, CA
gaoshi@cs.ucla.edu

Carlo Zaniolo
CSD, UCLA
Los Angeles, CA
zaniolo@cs.ucla.edu

ABSTRACT

Wikipedia's InfoBoxes play a crucial role in advanced applications and provide the main knowledge source for DBpedia and the powerful structured queries it supports. However, InfoBoxes, which were created by crowdsourcing for human rather than computer consumption, suffer from incompleteness, inconsistencies, and inaccuracies. To overcome these problems, we have developed (i) the *IBminer* system that extracts InfoBox information by text-mining Wikipedia pages, (ii) the *IKBStore* system that integrates the information derived by *IBminer* with that of DBpedia, YAGO2, WikiData, WordNet, and other sources, and (iii) SWiPE and InfoBox Editor (*IBE*) that provide a user-friendly interfaces for querying and revising the knowledge base. Thus, *IBminer* uses a deep NLP-based approach to extract from text a semantic representation structure called *TextGraph* from which the system detects patterns and derives subject-attribute-value relations, as well as domain-specific synonyms for the knowledge base. *IKBStore* and *IBE* complement the powerful, user-friendly, by-example structured queries of SWiPE by supporting the validation and provenance history for the information contained in the knowledge base, along with the ability of upgrading its knowledge when this is found incomplete, incorrect, or outdated.

1. INTRODUCTION

Knowledge bases (KBs) are playing a crucial role in many systems, such as text summarization and classification, opinion mining, semantic search, and question answering systems. In recent years, several projects have been devoted to create such KBs [2, 4, 5, 10, 11, 12, 13, 23, 25]. Many of these KBs are based on the structured summaries in Wikipedia, called *InfoBoxes*; each InfoBox summarizes important properties and their values for the entity (subject) described in the Wikipedia's page containing the InfoBox.

In addition to being very valuable for human readers, InfoBoxes and KBs have shown a significant potential of bringing us closer to the realization of the *Semantic Web* vision. For instance, Figure 1 shows the InfoBox for the UCLA Law School. The information harvested from Wikipedia InfoBoxes like this has been stored into RDF-based KBs (e.g., DBpedia), which support powerful *SPARQL* queries. Thus, queries that seek law schools satisfying simple (e.g., 'Bar pass rate' $> 90\%$) can now be expressed in *SPARQL*, along with more complex queries requiring join operations (e.g., law school and business school at the same university), or decision-support aggregates (e.g., law schools with the best faculty/students ratio). This new capability paves the way to powerful Semantic Web applications, but is confronted by the two main obstacles which are discussed next.

UCLA School of Law

Motto	*Fiat lux* (Latin)
Parent school	University of California
Established	1949
School type	Public
Parent endowment	$1.88 billion (June 30, 2009) [1]
Dean	Rachel Moran
Location	Los Angeles, California, U.S.
Enrollment	1,011[2]
Faculty	116–138[2]
***USNWR* ranking**	17[3]
Bar pass rate	85%[2]
Website	www.law.ucla.edu
ABA profile	ABA Law School Data

Figure 1: InfoBox for the UCLA Law School

Ease of Access represents the first major problem, since the knowledge base is now usable only by people who can write *SPARQL* queries—thus casual users are excluded. Even expert programmers will need to spend a fair amount of time to learn DBpedia and thousands of names of entities and properties there used (e.g., names such as: `foaf:givenName` and `dbpprop:populationTotal`). Much progress has been made on the ease-of-access front with the introduction of SWiPE [8] that uses a Query-By-Example (QBE) approach on InfoBoxes treated as

input query forms. For instance, to find law schools with certain properties in Wikipedia, a user will start from the InfoBox of a familiar school (such as that in Figure 1) and replace the existing values of the desired properties with conditions that specify the query. In Section 3, we provide an overview of the enhanced SWiPE which combines more powerful structured-query capabilities (e.g., joins and aggregates) with the free-text keyword-based retrieval capabilities of Web search engines.

Incompleteness and inconsistency represent major issues for current KBs. These problems are largely due to ad-hoc crowdsourcing being used to generate summaries, inasmuch as standard ontologies were either unavailable or ignored during this process. Currently, more than 40% of Wikipedia pages are missing their InfoBoxes entirely, and the others contain InfoBoxes that are often incomplete. A related problem is that when the information is present, it might be represented using synonymous attributes, such as '*birth date*', '*date of birth*', and '*born*', as it is in fact the case for DBpedia and many manually created KBs.

To address these challenges, we employ text-mining techniques to extract more knowledge from unstructured data and integrate knowledge from different knowledge sources. Thus, we have developed the following integrated systems:

IBminer [19] and *OntoMiner* [20, 21] (Section 4), which respectively build structured summaries and ontologies from document corpora using text-mining,

IKBStore and CS^3 [17] (Section 5) that integrate knowledge from different sources and realign subject and attribute names to construct a general KB having the quality and coverage needed for semantic applications, and

IBE [18] (Section 6) that supports the knowledge editing and query functions needed for managing and upgrading the KB, while maintaining provenance information on each piece of knowledge entered into the system.

We will now provide a high-level overview of these systems that provide an integrated set of user-friendly tools whereby the KB can be generated, unified, searched, and also upgraded while minimizing the expertise on KB terminology and system internals required from users.

2. SYSTEM ARCHITECTURE

Figure 2 shows the three-level architecture that supports the three functions of (i) Querying the KB, (ii) Knowledge Mining and (iii) Knowledge Integration and Management.

Querying the KB. We extended the original SWiPE [8] and its user-friendly QBE-like interface to support complex queries that combine joins, aggregates, and keyword-based searches. Thus, queries entered on the InfoBox

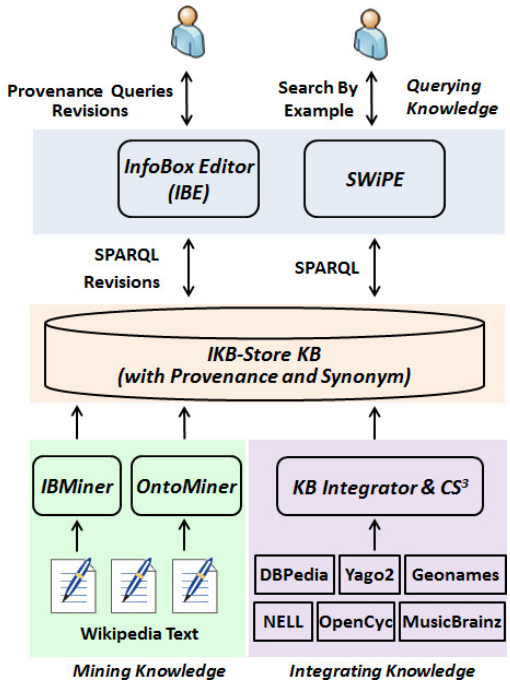

Figure 2: System Architecture

template are translated into SPARQL queries and executed on Virtuoso databases. On the other hand, *IBE* provides a simple interface for upgrading and managing the integrated KB, and also supports temporal and provenance queries on derived information.

Mining Knowledge. To improve coverage of our integrated KB, we developed the *IBminer* and the *OntoMiner* systems, which generate InfoBoxes and ontologies from free text. *IBminer* employs an NLP-based text mining engine called *SemScape* [22] to identify the morphological information in text and generate graph-based structures called TextGraphs. Then, *IBminer* extracts semantic links from TextGraphs using predefined patterns and converts semantic links to final InfoBoxes. In a fashion akin to *IBminer*, *OntoMiner* uses graph pattern rules to mine iteratively ontological information from text.

Integrating Knowledge. To integrate the knowledge from different sources into *IKBStore*, we use the existing interlink information from DBpedia. However, many subjects of interest are not covered by existing links, and links between corresponding attribute names (i.e., synonyms) are limited to the internal ones provided by DBpedia. Thus, the Context-Aware Synonym Suggestion System (CS^3) [17] was developed to discover context-aware synonyms for attributes and subjects. Furthermore, we built the *IBE* system which supports direct auditing and editing of the integrated KB by its curators. Thus, *IBE* was first used to address inconsistencies and other issues that surfaced during the integration of knowledge from different sources previously described,

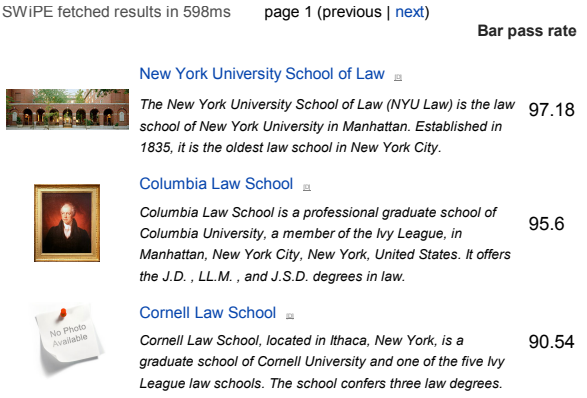

Figure 3: *SWiPE* result page

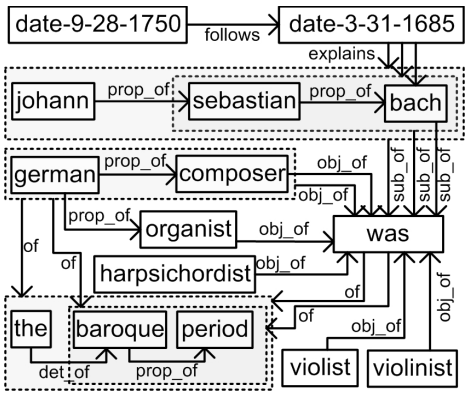

Figure 4: Part of the TextGraph for our example.

and it is currently used to manage our KB and upgrade it with information generated by crowdsourcing. Ease of access is achieved if the system can take users' queries expressed in natural language and translate them into *SPARQL* queries [14]. As discussed in [15], good results can be obtained for simple short queries; in fact, voice input can also be used in very simple ambiguity-free requests. These approaches however cannot handle complex queries, and even for simple ones the risk of misinterpretation is high.

3. BY-EXAMPLE STRUCTURED QUERY

We advocate the use of the ambiguity-free By-Example Structured Query (BEStQ) approach of *SWiPE* [8], that allows users to express with ease a large subset of the queries expressible in *SPARQL* [9]. In *SWiPE*, InfoBoxes are made active and users can enter conditions and constraints on each InfoBox item in a way similar to *Query-By-Example* (QBE). For example, to find in Wikipedia law schools with certain properties, a user starts from the InfoBox of any law school (e.g., that in Figure 1) and replaces the existing values of the desired properties with query conditions. For instance, the user could specify $>$90% for the '`Bar pass rate`' and '`New York`' in the `Location` field indicating the *State*. Then *SWiPE* translates this two-line BEStQ specification into the equivalent (22-line long) *SPARQL* query, and executes it on DBpedia using Virtuoso; results returned by Virtuoso are reformatted by *SWiPE* and presented to the user (Figure 3).

The original *SWiPE* prototype described in [8] was recently extended with more powerful structured-query capabilities based on joins, aggregates, and closure properties, and with the keyword-based retrieval capabilities of free-text search engines. The integration of BEStQ and keyword search is made possible by the fact that *SWiPE* displays the original pages from Wikipedia which contain a search box in the right top corner. Thus, in addition to the two conditions previously entered in the InfoBox, our *SWiPE* user could, e.g., enter the words '`ivy league`' in the search box. Then, *SWiPE* will return all Wikipedia pages satisfying both the conditions in the InfoBox and those in the search box. Therefore, the first entry in Figure 3 will be omitted, since 'New York University' is not in the 'Ivy League'. This simple example illustrates the dramatic improvements in precision and recall delivered by BEStQ searches and their synergy with more traditional keyword-based searches. This becomes obvious, if we try to express this query using only keywords and still get reasonable recall and precision[1]. While the integration of BEStQ and keyword search reaches new levels of precision and recall, our goal of producing high-quality answers cannot be reached unless we also tackle the incompleteness and inconsistency problem of the underlying KB. This is the focus of the rest of the paper.

4. KNOWLEDGE FROM TEXT

IBminer [19] and *OntoMiner* [20, 21], described in this section, use a deep NLP-based knowledge extraction approach to improve the completeness, consistency, and accuracy of the KB. The generation of TextGraphs and semantic links from text (Subsection 4.1), and the techniques that (i) learn patterns from TextGraphs and existing KBs and (ii) use them to generate InfoBox triples are discussed in Subsections 4.2, and 4.3. Subsection 4.4 covers *OntoMiner* and text-mining for ontological information.

4.1 TextGraphs and Semantic Links

The first step of knowledge extraction is to convert the sentences of the document into weighted graphs, which

[1]The best keyword combination we found to emulate our previous query is: `Ivy League Law Schools New York bar pass rate`; on this, Wikipedia returns a total of 46 answers. While this represents a major improvement w.r.t. the 1,040,000 results found by Google search, most of those 46 answers are irrelevant or outright ridiculous, such as "List of Batman: The Brave and the Bold Characters."

are called *TextGraphs*. This step is performed using *SemScape*, which uses Co-reference Resolution and other novel techniques to generate high-quality TextGraphs [22]. TextGraphs generated in this way provide a semantic representation of the grammatical connections between words, terms, and phrases through labeled and weighted links. For instance, Figure 4 shows the TextGraph for following sentence:

Example Sentence: "*Johann Sebastian Bach (31 March 1685 – 28 July 1750) was a German composer, organist, harpsichordist, violist, and violinist of the Baroque Period.*"

Once TextGraphs are generated, we use a set of predefined graph rules [6] to produce *Semantic Links* between concepts and terms in TextGraphs. For instance, the following *SPARQL*-like rule is used to produce <*Johann Sebastian Bach*, *was*, *composer*> and similar semantic links:

SELECT (?1 ?3 ?2)
WHERE { ?1 "subj_of" ?3. ?2 "obj_of" ?3.
NOT("not" "prop_of" ?3).
NOT("no" "det_of" ?1).
NOT("no" "det_of" ?2). }

4.2 Learning InfoBox Patterns

To map the semantic links generated in the previous subsection to the standard InfoBox triples, we learn patterns from the matching examples. For instance, consider the two semantic links <*bach*, *was*, *composer*> and <*bach*, *was*, *German*> generated from the TextGraph in Figure 4. Obviously, the link name '*was*' should be interpreted differently in these two cases, since the former is connecting a '*person*' to an '*occupation*', while the latter is between a '*person*' and a '*nationality*'. Now, consider the two InfoBox items <*bach*, *occupation*, *composer*> and <*bach*, *nationality*, *German*> which respectively match the mentioned triples from the text. These items clearly indicate that the link name '*was*' in our two triples should be interpreted respectively as '*occupation*' and '*nationality*'. Thus, from these examples, we learned the following two patterns (also called *Potential Matches* or *PMs*):

- <*cat:Person*, *was*, *cat:Occupation_in_Music*>: *occupation*
- <*cat:Person*, *was*, *cat:German*>: *nationality*

Here the PM $< c_1, l, c_2 >: \alpha$ indicates that the link named l, connecting a subject in category c_1 to a subject or value in category c_2, can be interpreted as the attribute name α. Observe that, instead of using the actual subjects and values in PMs, we used their categorical information to create more general and useful patterns. As a result, for each triple with a matching InfoBox item, we create several patterns since subjects and values usually belong to more than one (direct or indirect) categories. *IBminer* also counts the number of times each PM has occurred.

4.3 Generating Structured Summaries

To extract new structured summaries using PMs, for a given semantic link, say $<s, l, v>$, *IBminer* finds all potential matches such as $<c_s, l, c_v>: \alpha_i$. The resulting set of potential matches are then grouped by the InfoBox attribute names, α_i's. For each group, *IBminer* computes the aggregate frequency of the matches (called *evidence count*). At this point, *IBminer* removes infrequent potential matches using a threshold on normalized frequency, and then applies a type-checking to eliminate implausible matches [16]. Finally the matches remaining in the list are *ranked* according to their evidence count. The match with the largest evidence count, pm, is selected as a new InfoBox tuple $<t_n.s, pm.a_i, t_n.v>$ with confidence $t_n.c \times pm.c$ and evidence $t_n.e$. More details are provided in [16, 19].

4.4 Generating Ontological Information

The *OntoMiner* system uses a successive refinement approach to generate ontological information from text. It performs successive passes, where each pass consists of (i) a relation extraction phase which generates ontological relations between the existing terms and (ii) the concept extraction phase that detects new concepts and aliases using the generated relations. At each pass *OntoMiner* transforms each sentence into a TextGraph assuming the current ontology. New ontological relations between nodes are extracted using predefined graph rules (similar with the ones for generating semantic links). Then, it combines the generated relations, creating a list of ontological relations between terms with their weight and frequency. In the concept extraction phase, *OntoMiner* uses ontological relations between concepts and non-concept terms to detect new concepts and aliases that can be used to enhance the current ontologies. This step can also be performed under the supervision of a human who decides if another cycle of this process is needed or we can stop with the new and improved ontologies.

5. KNOWLEDGE INTEGRATION

To construct a comprehensive KB, we take the knowledge gathered from various public sources and that harvested by our knowledge extraction systems, and integrate these inputs into a KB seeking to achieve superior quality and coverage. A serious obstacle that limits the quality of the result is that different systems do not use a standard terminology, and often describe the same concept or attribute by different names. In this section, we introduce the Context-Aware Synonym Suggestion System (CS^3) which generates synonyms for both subjects and attributes whereby KBs are combined using these synonyms along with other interlinks.

5.1 Generating Synonyms

Different KBs use different terminologies for naming their attributes, and non-unique attribute names might also be used in the same KB. For instance in *DBpedia*, the attribute names '*birthdate*', '*data of birth*', '*born*' are all used to indicate the birthdate of a person, whereas *YAGO2* typically uses '*wasBornOnDate*' to refer to birthdate. Moreover, the attribute name '*born*' is used for both birthdate and birth-place in many systems. Indeed, synonyms are the source of significant ambiguities and inconsistencies.

To address this problem, our systems learn synonym patterns from TextGraph triples [16, 17]. Formally, if TextGraph triple $<s, l, v>$ matches patterns $<c_s, l, c_v>:\alpha_1$ and $<c_s, l, c_v>:\alpha_2$ respectively with evidence frequency f_1 and f_2 ($f_1 \geq f_2$), we create the following *Potential Attribute Synonym* (*PAS*) pattern: $<c_s, \alpha_1, c_v>: \alpha_2$.

This synonym pattern indicates that attribute name α_1 from subject category c_s to value category c_v is also known as α_2. Again, less frequent PAS patterns and those with low support are filtered out and the rest is used to suggest attribute interlinks for existing or newly generated InfoBoxes. Similar techniques are used for learning subject synonyms [16, 17].

5.2 Combining Knowledge Bases

Our first step in building IKBstore consisted in integrating the knowledge extracted from domain-specific KBs, such as MusicBrainz [4] and Geonames [2], and from several general-purpose ones, such as DBpedia [10] and YAGO2 [13]. Every piece of information in our system is represented by an RDF triple $<$*subject*, *attribute*, *value*$>$.

The naively integrated KB so obtained contains many synonymous terms and duplicate triples. Thus we utilize the interlinks provided by existing KBs and synonyms generated by CS^3 to improve the consistency of this initial integrated KB, using the techniques described next.

Interlinking Subjects: Fortunately, DBpedia has linked its subjects with other public KBs on the Web. We exploit existing interlinks in DBpedia to combine the information on the same subject from difference sources. For the cases lacking such interlinking information (e.g., NELL [12]), in addition to exact matching, we use synonym and context matching. Synonyms can be obtained from *redirect* and *sameAs* links in DBpedia, WordNet [24], CS^3, and OntoMiner. As for the context, we view identical attributes and values for different subjects as indication of possible matchings.

Interlinking Categories: In addition to exact matching, we compute the similarity of the categories in different KBs on the basis of their instances. Consider two categories c_1 and c_2, and let $S(c)$ be the set of subjects in category c. The similarity function for categories interlink is defined as $Sim(c_1, c_2) = |S(c_1) \cap S(c_2)|/|S(c_1) \cup S(c_2)|$. If the $Sim(c_1, c_2)$ is greater than a certain threshold, we consider c_1 and c_2 as aliases of each other.

After semantic integration, IKBstore contains 9.2 million English subjects and 105.4 million triples. All triples in our integrated KB are assigned accuracy, confidence, and frequency values, as explained in [19]. The sources from which the triples are generated are also stored to support provenance auditing in our KB.

6. INFOBOX EDITOR (IBE)

IBE [18] provides a user-friendly interface to manage InfoBox information while maintaining the provenance of this information. Therefore, *IBE* supports several important functions that streamline the crowdsourcing and management of InfoBox information, including:

1. Resolving inconsistencies in the knowledge collected from various KBs,
2. Extracting knowledge from user provided text by using *IBminer*,
3. Revising the InfoBoxes generated and adding new InfoBoxes to existing subjects,
4. Retrieving and revising meta-level information, such as ontologies and synonyms,
5. Recording the provenance history of InfoBoxes,
6. Supporting simple queries on current KB and its provenance history, along with transaction time queries to flash-back to the past.

While *IBE* is still a work-in-progress (only functions 1–4 from the list are currently implemented), it outlines a new level of functionality required for curated Web corpora to take a central role in advanced applications. Thus the new responsibility of curators will go beyond that of enabling the creation of textual documents and supervising their contents. They will also be responsible for promoting and supervising the process of knowledge creation and integration, crucial for the many applications that rely on the KBs created from Web document corpora. Recent developments, including Wikidata [7], underscore the significance of this trend.

7. EXPERIMENTS AND RESULTS

Extensive experiments were conducted to evaluate the effectiveness of our systems [6, 16, 18, 19, 21, 9]. In this section, we present the results we obtained by applying *IBminer* onto the text of the entire English Wikipedia, which is a corpus containing 4.4 Million subjects each described by 18.2 sentences on average.

The experiments took several weeks on the Hoffman2 cluster at UCLA [3] using up to 256 cores each with 8GB of main memory.

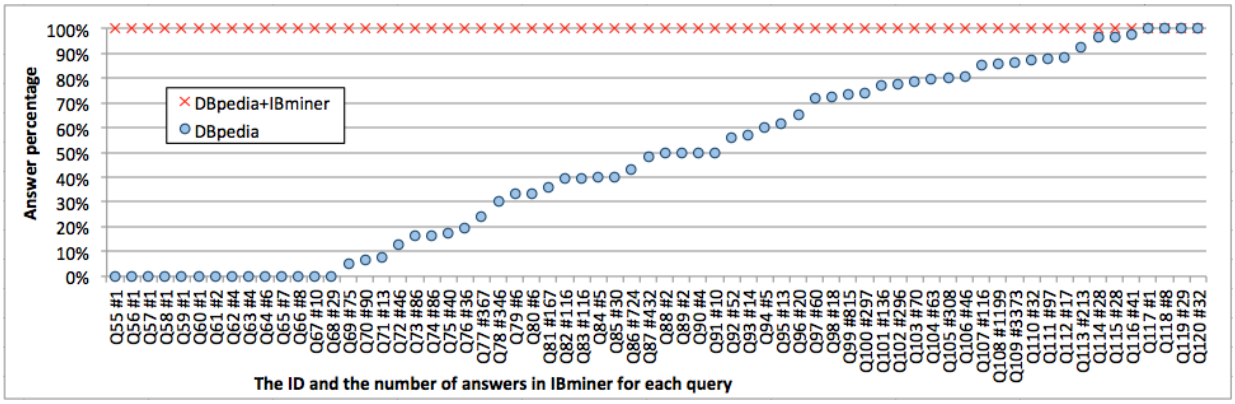

Figure 5: Number of results generated for popular queries using DBpedia and *IBminer*+DBpedia.

The use of *IBminer* on the entire English Wikipedia produced 251 Millions of links where subjects match their page titles. We ranked these links according to their confidence values that measure their supporting evidence (whereas many other links where the subject does not match the page title were simply discharged). In order to decide on the minimum confidence required for the links to be added to our KB, and produce InfoBox triples, we executed the following manual evaluation. We selected 50,000 of such triples and matched them against the text in the page. Therefore for our recall evaluation, we only used those InfoBox triples which match at least one of our semantic links, and measured the recall by computing how many of them are also generated by *IBminer*, whereas precision is measured by the percentage of such triples that are correct. As we lower the confidence threshold required, we obtain a larger recall but a worse precision. For instance, *IBminer* was able to generated 3.9 Million triples with 95% precision. When the threshold was lowered to 90% precision, we are able to recall 7.1 millions of new InfoBox triples.

A more meaningful way to evaluate the usefulness and completeness of a KB is to evaluate the degree of recall it entails for frequently used queries. The next experiment, akin to those performed in [14] and [15], attempts to evaluate the improvement obtained for popular queries on musicians and actors, when the 7.1 Million triples generated by *IBminer* are used.

Popular Queries: In order to create a set of popular queries for our evaluation, we used Google Search AutoComplete system, and found around 150 keyword queries suggested by this system to complete two phrases: "*musicians who*" and "*actors who*". We were able to translate 120 of these keyword-based queries to *SPARQL*. The remaining keyword queries, such as "*Actors who are tall*", "*Musicians who married normal people*", are too vague for a precise translation and quantification and were thus ignored.

Knowledge Bases: Two different KBs are used in this evaluation. As for the baseline KB, we use DBpedia's InfoBox triples. Since the goal is to measure how much *IBminer*'s result improves DBpedia, we combine the triples in DBpedia and *IBminer* into our second KB called IBminer+DBpedia.

After preparing the queries and the KBs, we employed Apache Jena [1], and ran the queries using the two KBs. For more than 44% of the queries, no answer is found from any of the KBs. This very clearly illustrates the incompleteness of the current KBs. Nevertheless, for the remaining queries, *IBminer*+DBpedia produces more answers than the baseline for all queries except four queries in which the additional triples of *IBminer* produced no additional result. Figure 5 shows what portion of the answers found using IBminer+DBpedia are found using only DBpedia's KB. The queries producing no answer are omitted from the figure, where the others are shown sorted by increasing percentages. The number of results found using IBminer+DBpedia is included in the horizontal axis under the query ID [6]. Figure 5 shows that for 11.6% of the queries for which DBpedia is not able to provide any answer, *IBminer* is actually able to find between 1 to 29 answers. Therefore while *IBminer* improves DBpedia's size (coverage) by 21.3%, it improves the completeness of the answers for popular queries by 53.3%.

8. CURRENT WORK & CONCLUSION

In this paper, we have described a set of tools for distilling Web corpora into knowledge bases that will enable structured queries on text documents, making it possible to support more advanced Semantic-Web applications. It is therefore clear that curators of Web corpora must take on responsibilities that go well beyond enabling access to Web documents and supervising their contents. Indeed curator groups must take on the role of KB managers who are responsible for promoting and

supervising the creation of computer-processable document summaries that will support sophisticated Semantic-Web applications and powerful structured queries, such as those provided by *SWiPE* [8].

The importance of curated or semi-curated Web corpora, featuring integrated, well-managed knowledge bases, is underscored by the success of Wikipedia and Wikidata [7] which is revising, completing, and improving current InfoBoxes with interlingual links. These developments represent great news for our project: in fact according to our plan of future work, *IKBStore* will be extended and improved with knowledge taken from Wikidata and Freebase, and *SWiPE* will be extended with multilingual query capabilities. Moreover, we plan to apply our tools and approach to document corpora other than Wikipedia — e.g., medical and technical encyclopedias. For many of these applications, the massive crowdsourcing approach of Wikipedia will not be cost-effective, and our approach that relies on text-mining and various semi-automatic tools will be more desirable.

9. ACKNOWLEDGEMENT

The authors would like to thank the reviewers and the editors for several suggested improvements. This work was partially supported by NSF Grant IIIS-1118107, RAS project CRP-17615 *DENIS: Dataspaces Enhancing Next Internet in Sardinia*, and MIUR PRIN 2010-11 project *Security Horizons*.

10. REFERENCES

[1] Apache Jena. http://jena.apache.org/.
[2] Geonames. http://www.geonames.org/.
[3] Hoffman2 Cluster, UCLA. http://hpc.ucla.edu/hoffman2/.
[4] Musicbrainz. http://musicbrainz.org/.
[5] Opencyc. http://www.cyc.com/platform/opencyc .
[6] Semantic web information management system (swims). http://semscape.cs.ucla.edu/.
[7] Wikidata. http://www.wikidata.org.
[8] M. Atzori and C. Zaniolo. Swipe: searching wikipedia by example. In *WWW (Companion Volume)*, pages 309–312, 2012.
[9] M. Atzori and C. Zaniolo. Expressivity and accuracy of by-example structure queries on wikipedia. CSD Technical Report #140017, UCLA, 2014.
[10] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia - a crystallization point for the web of data. *J. Web Sem.*, 7(3):154–165, 2009.
[11] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.
[12] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.
[13] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artif. Intell.*, 194:28–61, 2013.
[14] R. Huang and L. Zou. Natural language question answering over rdf data. In *SIGMOD Conference*, pages 1289–1290, 2013.
[15] Lei Zou et al. Natural language question answering over rdf: a graph data driven approach. In *SIGMOD Conference*, pages 313–324, 2014.
[16] H. Mousavi. *Summarizing Massive Information for Querying Web Sources and Data Streams*. PhD thesis, UCLA, 2014.
[17] H. Mousavi, S. Gao, and C. Zaniolo. Discovering attribute and entity synonyms for knowledge integration and semantic web search. *3rd International Workshop on Semantic Search over The Web*, 2013.
[18] H. Mousavi, S. Gao, and C. Zaniolo. Ibminer: A text mining tool for constructing and populating infobox databases and knowledge bases. *PVLDB*, 6(12):1330–1333, 2013.
[19] H. Mousavi, D. Kerr, M. Iseli, and C. Zaniolo. Deducing infoboxes from unstructured text in wikipedia pages. In *CSD Technical Report #130001), UCLA*, 2013.
[20] H. Mousavi, D. Kerr, M. Iseli, and C. Zaniolo. Ontoharvester: An unsupervised ontology generator from free text. In *CSD Technical Report #130003), UCLA*, 2013.
[21] H. Mousavi, D. Kerr, M. Iseli, and C. Zaniolo. Harvesting domain specific ontologies from text. In *ICSC*, 2014.
[22] H. Mousavi, D. Kerr, M. Iseli, and C. Zaniolo. Mining semantic structures from syntactic structures in free text documents. In *ICSC*, 2014.
[23] P. Singh, T. Lin, E. T. Mueller, G. Lim, T. Perkins, and W. L. Zhu. Open mind common sense: Knowledge acquisition from the general public. In *Confederated International Conferences DOA, CoopIS and ODBASE*, London, UK, 2002.
[24] M. M. Stark and R. F. Riesenfeld. Wordnet: An electronic lexical database. In *Proceedings of 11th Eurographics Workshop on Rendering*. MIT Press, 1998.
[25] W. Wu, H. Li, H. Wang, and K. Q. Zhu. Probase: a probabilistic taxonomy for text understanding. In *SIGMOD Conference*, 2012.

The Database Group at TUM

www-db.in.tum.de

Alfons Kemper
Technische Universität München
kemper@in.tum.de

Thomas Neumann
Technische Universität München
neumann@in.tum.de

The database group at TUM was established in 1972 by Rudolf Bayer [4] who retired in 2004. The group is reponsible for the foundational and advanced database curriculum of the department's 3500 undergraduate and graduate students. The group consists of two post-docs and around 15 doctoral students. In the following we will overview the research agenda of the last couple of years only; previous projects and publications can be viewed on our web site.

1. THE HYPER PROJECT

Most of the recent work of the TUM database group was done within the context of the long-term HyPer project. The goal of this project is to develop a high-performance database engine that (finally) unites the two seemingly disparate worlds: OLTP and OLAP. For this purpose we developped a hybrid main memory database system that supports OLTP- **and** OLAP-applications – in parallel on the same database state. The key idea is to exploit the OS/processor-support for virtual memory management. This allows to spawn consistent database snapshots to isolate OLAP queries from the OLTP transactions – even though they share the same database [13]. After about three years of development, HyPer now is a fairly mature database system and can be experimented with by other researchers via our web site `hyper-db.de` as shown in Figure 1.

In HyPer the OLTP process "owns" the database and periodically (e.g., in the order of seconds or minutes) forks an OLAP process. This OLAP process constitutes a fresh transaction consistent snapshot of the database. Thereby, we exploit operating systems functionality to create virtual memory snapshots for new, cloned processes. In Unix, for example, this is done by creating a child process of the OLTP process via the `fork` system call.

The forked child process obtains an exact copy of the parent processes address space, as exemplified on the lower right-hand side in Figure 1 by the overlayed page frame panel. This virtual memory snapshot that is created by the `fork`-operation will be used for executing a session of OLAP queries. These queries can be executed in parallel threads or serially, depending on the system resources or client requirements. In essence, the virtual memory snapshot mechanism constitutes a OS-/hardware supported shadow paging mechanism as proposed decades ago for disk-based database systems by Lorie [17]. However, the original proposal incurred severe costs as it had to be software-controlled and it destroyed the clustering on disk. Neither of these drawbacks occurs in the virtual memory snapshotting as clustering across RAM pages is not an issue. Furthermore, the sharing of pages and the necessary copy-on-update/write is managed by the operating system with effective hardware support of the MMU (memory management unit) via the page table that translates VM addresses to physical pages and traps necessary replication (copy-on-write) actions.

Our performance results demonstrate that HyPer combines the best of the two "worlds": HyPer's OLTP performance is comparable to that of dedicated OLTP engines (like VoltDB or Hekaton) and HyPer's OLAP query response times match those of the best pure OLAP engines (e.g., MonetDB and VectorWise). It should be emphasized that HyPer can match (or beat) these two best-of-breed transaction (VoltDB) and query (MonetDB, VectorWise) processing engines **at the same time** by performing both workloads in parallel on the same database state. This performance evaluation was carried out on the basis of a new business intelligence benchmark, the so-called CH-benCHmark [7], that combines the transactional workload of TPC-C with the OLAP queries of TPC-H – executed against the **same** database state. Hyper's excellent performance is due to the following design choices:

- HyPer relies on in-memory data management without the ballast of traditional database systems that is caused by DBMS-controlled page structures and buffer management. The SQL table definitions are transformed into simple vector-based virtual memory representations – which constitutes a column-oriented physical storage scheme.

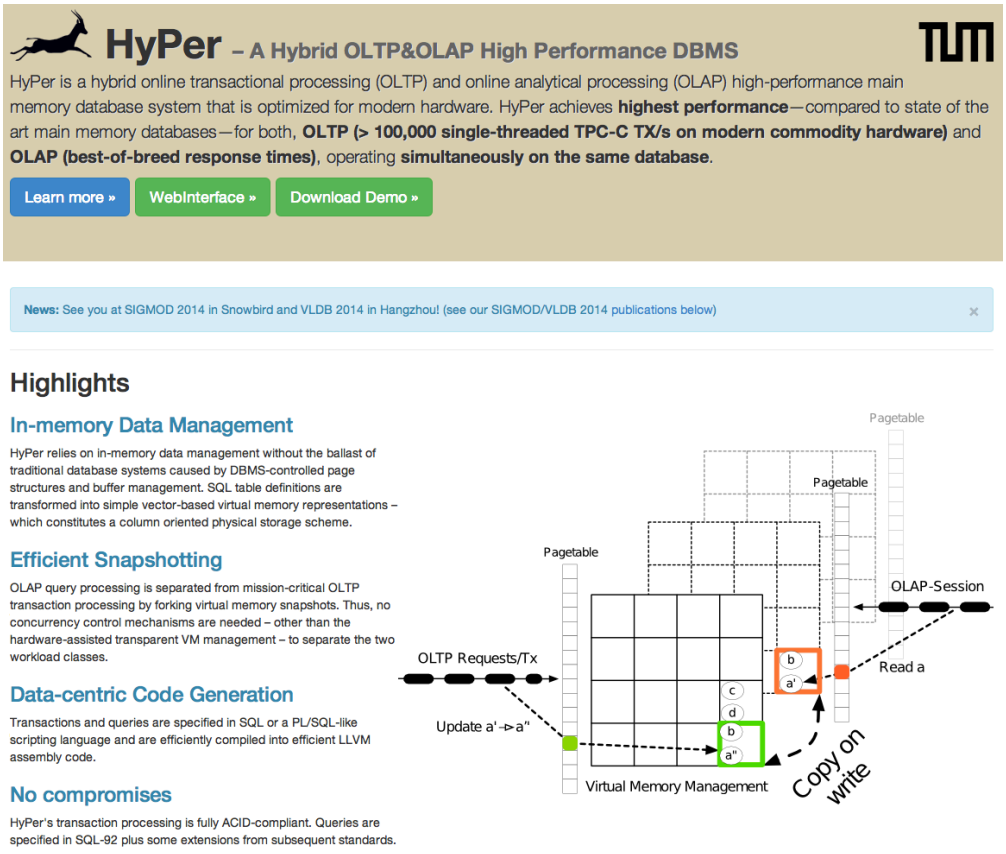

Figure 1: `www.hyper-db.de` Web Site

- The OLAP processing is separated from the mission-critical OLTP transaction processing by **forking** virtual memory snapshots. Thus, no concurrency control mechanisms other than the hardware-assisted VM management are needed to separate the two workload classes.
- Transactions and queries are specified in SQL and are efficiently compiled into LLVM assembly code [21]. The transactions are specified in an SQL scripting language (called HyPer-Script) and registered as precanned, stored procedures. For the JIT-compiled interactive queries the very fast compilation into assembly language (LLVM) as opposed to a slow cross-compilation (into C/C++) is essential. The query evaluation follows a data-centric paradigm by applying as many operations on a data object as possible in between pipeline breakers. This evaluation scheme goes one step beyond cache-locality towards register-locality.
- The lock-free execution model [16, 18] in combination with group committing achieves extreme scalability in terms of transaction throughput – without compromising the "holy grail" of ACID (that was sacrificed by the NoSQL/key value stores).
- While in-core OLAP query processing can be based on sequential scans, this is not possible for transaction processing. Therefore, we have developed a sophisticated main-memory indexing structure, the ART tree [15].
- We developed an extremely efficient loading process [19] that allows to use HyPer for big data exploration by loading data a window at a time for deep data analysis.
- For efficiently scaling out the HyPer database engine we developed a locality-sensitive query engine [23].
- The index structure DeltaNI [10] was developed to support hierarchical data in main-memory database systems.
- HyPer has a very small memory footprint which allows to run the same system on both, brawny nodes with up to several TB of main-memory and a hundred cores as well as on wimpy nodes, such as tablets and smartphones [20].

Considering that it is an active research project the implementation of HyPer is fairly mature. Even though it is not open source, we make it publicly available for experimentation via our web site `hyper-db.de`. We

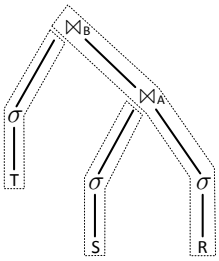

Figure 2: An Execution Plan with visible Pipeline Fragments for a Three-Way-Join

also use it for teching purposes; e.g., our 750 students of the lecture "Foundations of Database Systems" use it for learning SQL programming and query optimizer experiments.

2. OVERVIEW OF RECENT WORK

2.1 Efficient Query Compilation

HyPer uses a lock-free execution model. This assumes that transactions, in particular OLTP transactions, are very fast. HyPer achieves this by using LLVM for just-in-time compilation of SQL queries to machine code [21]. The compilation works in a data-centric manner: Instead of compiling one operator at a time, the compiler generates code for each individual data pipeline of the execution plan. This is illustrated in Figure 2. The pipelines, i.e., the data flow paths from one materialization point to the next, are compiled in a bottom-up manner where every operator pushes tuples towards its consumer. In code this means that most pipeline fragments consist of a few tight loops, which is favorable for modern CPUs and results in excellent performance. This compilation step avoids the high CPU overhead of classical interpreted execution frameworks. For disk-based systems this overhead was largely neglectable, but for in-memory processing any interpretation overhead is very visible.

2.2 NUMA-Aware Many-Core Parallelism

The main impetus of hardware performance improvement nowadays comes from increasing multi-core parallelism rather than from speeding up single-threaded performance. Intel's new mainstream server Ivy Bridge EX (that we recently obtained) can run 120 concurrent threads in a 4-socket configuration. We use the term *many-core* for such architectures with tens or hundreds of cores.

At the same time, increasing main memory capacities of up to several TB per server have led to the development of main-memory database systems. In these systems query processing is no longer I/O bound, and the huge parallel compute resources of many-cores can be truly exploited. Unfortunately, the trend to move mem-

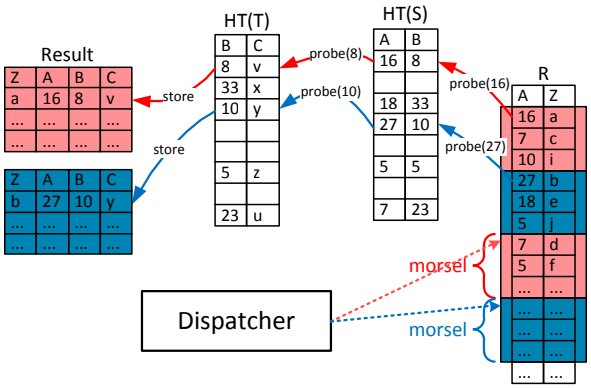

Figure 3: Idea of morsel-driven parallelism: $R \bowtie_A S \bowtie_B T$

ory controllers into the chip and hence the decentralization of memory access, which was needed to scale throughput to huge memories, leads to non-uniform memory access (NUMA). In essence, the computer has become a network in itself as the access costs of data items varies depending on which chip the data and the accessing thread are located. Therefore, many-core parallelization needs to take RAM and cache hierarchies into account. [1, 3] developed and analyzed massively parallel join algorithms. In HyPer [14] we extend this work to an effective end-to-end parallelization scheme covering entire query evaluation plans. In particular, the NUMA division of the RAM has to be considered carefully to ensure that threads work (mostly) on NUMA-local data.

HyPer employs an adaptive *morsel-driven* query execution framework – as described in [14]. Our approach is sketched in Figure 3 for the three-way-join query $R \bowtie_A S \bowtie_B T$. Parallelism is achieved by processing each pipeline on different cores in parallel, as indicated by the two (upper and lower) *probe-probe-store*-pipelines in the figure. The core idea is a *scheduling* mechanism (the "dispatcher") that allows flexible parallel execution of an operator pipeline, that can change the parallelism degree even during query execution. A query is divided into segments, and each executing segment takes a morsel (typically 100,000) of input tuples and executes these, materializing results in the next pipeline breaker. The morsel framework enables NUMA local processing as indicated by the color coding in the figure: A thread operates on NUMA-local input and writes its result into a NUMA-local storage area. Our dispatcher runs a fixed, machine-dependent number of threads, such that even if new queries arrive there is no resource oversubscription, and these threads are pinned to the cores, such that no unexpected loss of NUMA locality can occur due to the OS moving a thread to a different core.

The crucial feature of morsel-driven scheduling is that task distribution is done at run-time and is thus *fully elastic*. This allows to achieve perfect load balancing, even in the face of uncertain size distributions of in-

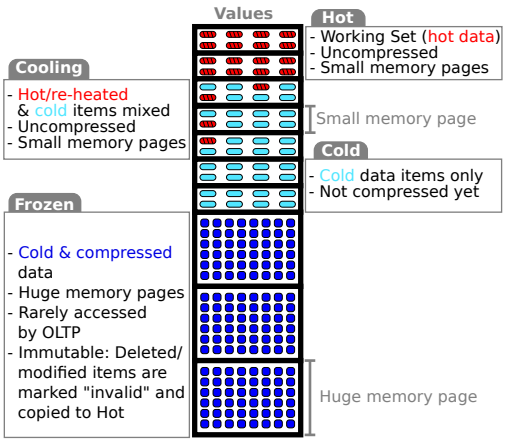

Figure 4: Hot/cold Clustering for Compression.

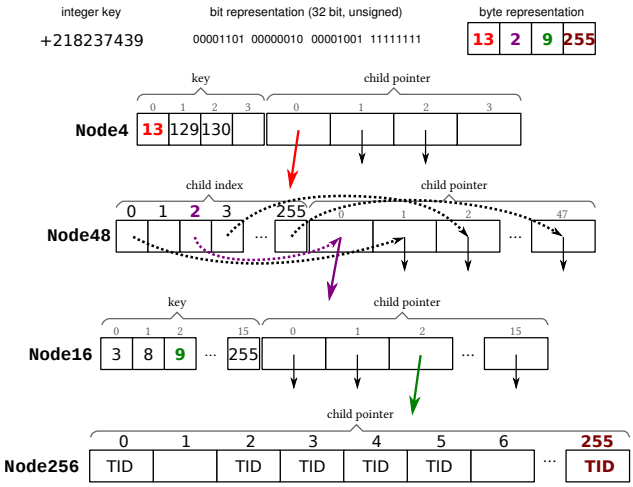

Figure 5: The Adaptive Radix Tree ART: Sample Path for the Key 218237439 Traversing all four Node Types

termediate results, as well as the hard-to-predict performance of modern CPU cores that varies even if the amount of work they get is the same. It is elastic in the sense that it can handle workloads that change at run-time (by reducing or increasing the parallelism of already executing queries in-flight) and can easily integrate a mechanism to run queries at different priorities.

2.3 Compacting the In-Memory Database

Our approach [11] to compression in hybrid OLTP & OLAP column stores is based on the observation that while OLTP workloads frequently modify the dataset, they often follow the working set assumption: Only a small subset of the data is accessed and an even smaller subset of this working set is being modified (cf. Figure 4). In business applications, this working set is mostly comprised of tuples that were added to the database in the recent past, as it can be observed in the TPC-C workload (cf. `www.tpc.org`).

Our system uses a lightweight monitoring component to observe accesses to the dataset and identify opportunities to reorganize data such that it is clustered into hot and cold parts. After clustering, the database system compresses cold chunks to reduce memory consumption and streamline queries. In future versions we will even stage the frozen parts to less expensive memory, e.g., to non-volatile memory NVM, SSD or disk storage media. Our concept of compaction has received a lot of attention and is currently being incorporated in Microsoft's Hekaton [9] as well as in HStore/VoltDB [8].

2.4 Radix Tree Indexing in Main-Memory Databases

The efficiency of transaction processing largely depends on which index structures are used, as exemplified by the first three *select*-statements of the *newOrder* implementation of the TPC-C benchmark. In main-memory, dictionary-like data structures supporting insert, update, and delete are often implemented as hash tables or comparison-based trees (e.g. self-balancing binary trees or B-trees). Hashing is usually much faster than a tree as it offers constant lookup time in contrast to the logarithmic behavior of comparison-based trees. The advantage of trees is that the data is stored in sorted order, which enables additional operations like range scan, minimum, maximum, and prefix lookup.

The radix tree, also known as trie, prefix tree, or digital search tree, is another dictionary-like data structure. In contrast to comparison-based structures, which compare opaque key values using a comparison function, radix trees directly use the binary representation of the key. Although radix trees are often introduced as a data structure for storing character strings, they can be used to store any data type by considering values as strings of bits or bytes.

The complexity of radix trees for insert, lookup, and delete is $O(k)$ where k is the length of the key. The access time is *independent* of the number of elements stored. Besides the length of the key, the height of a radix tree depends on the number of children each node has. For example, a radix tree with a fanout of 256 that stores 32 bit integers has a height of 4.

So far radix trees suffered from space underutilization problems as typically an array of 256 pointers was allocated for each node – even though some nodes might have a very low fan-out compared to others. Therefore, we developed in [15] the Adaptive Radix Tree (ART), which uses four different node types that can handle up to (i) 4, (ii) 16, (iii) 48, and (iv) 256 entries. Thereby, a good space utilization of ART-trees is guaranteed while still being able to achieve a maximum height of k for k-byte keys. That is, 32 bit integers are indexed with a tree

of height 4, 64 bit integers require height 8. These adaptive nodes are exemplified by the sample path for the 32-bit key 218237439 consisting of the the 4 byte-chunks 13&2&9&255 in Fig. 5. This path starts at the root, which happens to be of type *Node4*, and then covers the three other node types. The structural representation of these node types varies, as illustrated in the figure. In designing the node structure the trade-off between space utilization and intra-node search performance was taken into account.

Besides adaptive nodes, we employ other compression techniques that bound the worst-case space consumption per key/value pair to 52 bytes – even for arbitrarily long keys [15].

2.5 Lock-Free Synchronization via Hardware Transactional Memory

The upcoming hardware transactional memory (HTM) support in mainstream processors like Intel's Haswell appears to be a perfect fit for optimizing the emerging main-memory database systems. Transactional memory [12] is a very intriguing concept that allows for automatic atomic and concurrent execution of arbitrary code.

However, transactional memory is no panacea for transaction processing. First, database transactions also require properties like durability, which are beyond the scope of transactional memory. Second, at least the current hardware implementations of transactional memory are limited. For the Haswell architecture the read-/write sets have to fit into the L1 cache with a capacity of 32KB, which limits the scope of a transaction. Furthermore, HTM transactions may fail due to a number of unexpected circumstances like collisions caused by cache associativity, hardware interrupts, etc. Therefore, it does not seem to be viable to map an entire database transaction to a single monolithic HTM transaction. In addition, one always needs a "slow path" to handle the pathological cases (e.g., associativity collisions).

We therefore developed in [15] an architecture where transactional memory is used as a building block for assembling complex database transactions. Along the lines of the general philosophy of transactional memory we start executing transactions optimistically, using (nearly) no synchronization and thus running at full clock speed. By exploiting HTM we get many of the required checks for free, without complicating the database code, and can thus reach a much higher degree of parallelism than classical locking or latching. In order to minimize the number of conflicts in the transactional memory component, we carefully control the data layout and the access patterns of the involved operations, which allows to operate largely without explicit synchronization.

Because the maximum size of hardware transactions is limited, only a database transaction that is small can

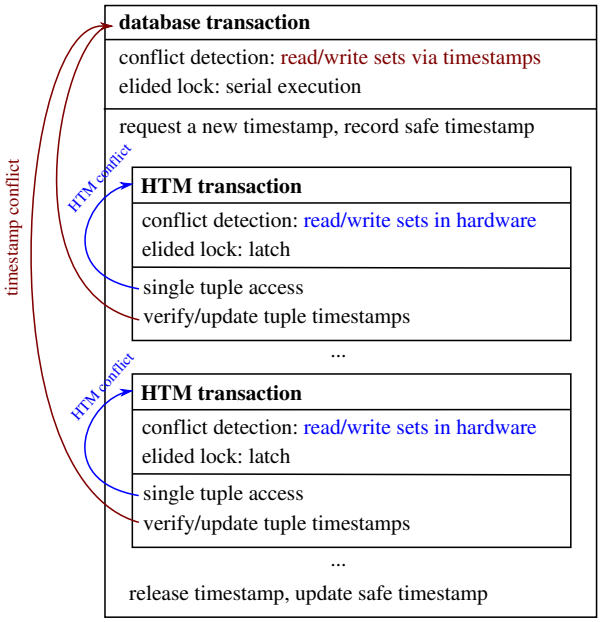

Figure 6: Transforming database transactions into HTM transactions

directly be mapped to a single hardware transaction. We therefore assemble complex database transactions by using hardware transactions as building blocks, as shown in Figure 6. The key idea here is to use a customized variant of timestamp ordering (TSO) to "glue" together these small hardware transactions. TSO is a classic concurrency control technique, which was extensively studied in the context of disk-based and distributed database systems [6, 5]. For disk-based systems, TSO is not competitive to locking because most read accesses result in an update of the read timestamp, and thus a write to disk. The timestamp updates are obviously much cheaper in RAM. On the opposite, fine-grained locking is much more expensive than maintaining timestamps in main memory, as we showed in [15].

Timestamp ordering uses read and write timestamps to identify read/write and write/write conflicts. Each transaction is associated with a monotonically increasing timestamp, and whenever a data item is read or updated its associated timestamp is updated, too. The *read timestamp* of a data item records the youngest reader of this particular item, and the *write timestamp* records the last writer. This way, a transaction recognizes if its operation collides with an operation of a "younger" transactions (i.e., a transaction with a larger timestamp), which would be a violation of transaction isolation. In particular, an operation fails if a transaction tries to read data from a younger transaction, or if a transaction tries to update a data item that has already been read by a younger transaction.

Lock-free execution is generally unsuitable for "ill-natured" transactions like long-running OLAP-style que-

ries or transactions querying external data – even if they occur rarely in the workload. In [18] we developed an approach whereby long transactions are first executed tentatively on the virtual memory snapshot and then applied to the main database as a short install transaction.

3. AWARDS

In the last couple of years the database group at TUM has obtained the following awards:

- The team "Campers" of Henrik Mühe and Florian Funke (mentored by Alfons Kemper and Thomas Neumann) was among the finalists of the ACM SIGMOD Programming Contest 2013 and achieved second place.
- The team "AWFY" of Moritz Kaufmann, Manuel Then, Tobias Mühlbauer, und Andrey Gubichev (mentored by Alfons Kemper and Thomas Neumann) won the ACM SIGMOD Programming Contest 2014.
- The ICDE 2014 Best Paper Award was presented to Viktor Leis, Alfons Kemper and Thomas Neumann for their paper "Exploiting Hardware Transactional Memory in Main-Memory Databases" [16].
- Thomas Neumann was awarded the Early Career Research Contribution Award at VLDB 2014 for his work on "Engineering High-Performance Database Engines" [22].

4. GUESTS

- Nikolaus Augsten (University of Salzburg) visited the group to work on similarity join processing [2].
- Peter Boncz (CWI/VU Amsterdam) was awarded the Humboldt Prize to work with us on multi-core parallel query processing [14].

5. VLDB 2017 AT TUM

The database group of TUM will organize the VLDB 2017 conference in Munich. It will take place during the last week of August 2017 at the main campus of TUM in downtown Munich.

6. ACKNOWLEDGMENTS

Our work is supported by various industry grants (SAP, Siemens, Oracle Labs, Google, IBM, HP Labs, Intel) as well as by the German Research Foundation DFG, the EU and the German Federal Ministry for Economic Affairs and Energy (BMWi).

7. REFERENCES

[1] M.-C. Albutiu, A. Kemper, and T. Neumann. Massively parallel sort-merge joins in main memory multi-core database systems. *PVLDB*, 5(10):1064–1075, 2012.
[2] N. Augsten, A. Miraglia, T. Neumann, and A. Kemper. On-the-fly token similarity joins in relational databases. In *SIGMOD Conference*, pages 1495–1506, 2014.
[3] C. Balkesen, G. Alonso, J. Teubner, and M. T. Özsu. Multi-core, main-memory joins: Sort vs. hash revisited. *PVLDB*, 7(1):85–96, 2013.
[4] R. Bayer. Kurzbiographie (in german). `http://www3.informatik.tu-muenchen.de/people/sites/bayer/Memoiren-Bayer.pdf`, 2014.
[5] P. A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, 1987.
[6] M. J. Carey. *Modeling and evaluation of database concurrency control algorithms*. PhD thesis, 1983.
[7] R. Cole, F. Funke, L. Giakoumakis, W. Guy, A. Kemper, S. Krompass, H. A. Kuno, R. O. Nambiar, T. Neumann, M. Poess, K.-U. Sattler, M. Seibold, E. Simon, and F. Waas. The mixed workload ch-benchmark. In *DBTest*, 2011.
[8] J. DeBrabant, A. Pavlo, S. Tu, M. Stonebraker, and S. B. Zdonik. Anti-caching: A new approach to database management system architecture. *PVLDB*, 6(14), 2013.
[9] A. Eldawy, J. J. Levandoski, and P.-Å. Larson. Trekking through siberia: Managing cold data in a memory-optimized database. *PVLDB*, 7(11):931–942, 2014.
[10] J. Finis, R. Brunel, A. Kemper, T. Neumann, F. Färber, and N. May. DeltaNI: an efficient labeling scheme for versioned hierarchical data. In *SIGMOD Conference*, 2013.
[11] F. Funke, A. Kemper, and T. Neumann. Compacting transactional data in hybrid OLTP & OLAP databases. *PVLDB*, 5(11):1424–1435, 2012.
[12] M. Herlihy and J. E. B. Moss. Transactional memory: Architectural support for lock-free data structures. In *ISCA*, 1993.
[13] A. Kemper and T. Neumann. HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots. In *ICDE*, 2011.
[14] V. Leis, P. Boncz, A. Kemper, and T. Neumann. Morsel-driven parallelism: A NUMA-aware query evaluation framework for the many-core age. In *SIGMOD*, 2014.
[15] V. Leis, A. Kemper, and T. Neumann. The adaptive radix tree: ARTful indexing for main-memory databases. In *ICDE*, pages 38–49, 2013.
[16] V. Leis, A. Kemper, and T. Neumann. Exploiting hardware transactional memory in main-memory databases. In *ICDE*, pages 580–591, 2014.
[17] R. A. Lorie. Physical integrity in a large segmented database. *ACM TODS*, 2(1), 1977.
[18] H. Mühe, A. Kemper, and T. Neumann. Executing long-running transactions in synchronization-free main memory database systems. In *CIDR*, 2013.
[19] T. Mühlbauer, W. Rödiger, R. Seilbeck, A. Reiser, A. Kemper, and T. Neumann. Instant loading for main memory databases. *PVLDB*, 6(14):1702–1713, 2013.
[20] T. Mühlbauer, W. Rödiger, R. Seilbeck, A. Reiser, A. Kemper, and T. Neumann. One DBMS for all: the brawny few and the wimpy crowd. In *SIGMOD Conference*, pages 697–700, 2014.
[21] T. Neumann. Efficiently compiling efficient query plans for modern hardware. *PVLDB*, 4(9), 2011.
[22] T. Neumann. Engineering High-Performance Database Engines. *PVLDB*, 7(13), 2014.
[23] W. Rödiger, T. Mühlbauer, P. Unterbrunner, A. Reiser, A. Kemper, and T. Neumann. Locality-sensitive operators for parallel main-memory database clusters. In *ICDE*, pages 592–603, 2014.

The Beckman Report on Database Research

Daniel Abadi, Rakesh Agrawal, Anastasia Ailamaki, Magdalena Balazinska, Philip A. Bernstein, Michael J. Carey, Surajit Chaudhuri, Jeffrey Dean, AnHai Doan, Michael J. Franklin, Johannes Gehrke, Laura M. Haas, Alon Y. Halevy, Joseph M. Hellerstein, Yannis E. Ioannidis, H.V. Jagadish, Donald Kossmann, Samuel Madden, Sharad Mehrotra, Tova Milo, Jeffrey F. Naughton, Raghu Ramakrishnan, Volker Markl, Christopher Olston, Beng Chin Ooi, Christopher Ré, Dan Suciu, Michael Stonebraker, Todd Walter, Jennifer Widom

Abstract

Every few years a group of database researchers meets to discuss the state of database research, its impact on practice, and important new directions. This report summarizes the discussion and conclusions of the eighth such meeting, held October 14-15, 2013 in Irvine, California. It observes that Big Data has now become a defining challenge of our time, and that the database research community is uniquely positioned to address it, with enormous opportunities to make transformative impact. To do so, the report recommends significantly more attention to five research areas: scalable big/fast data infrastructures; coping with diversity in the data management landscape; end-to-end processing and understanding of data; cloud services; and managing the diverse roles of people in the data life cycle.

1. INTRODUCTION

A group of database researchers meets periodically to discuss the state of the field and its key directions going forward. Past meetings were held in 1989 [BDD+89], 1990 [SSU91], 1995 [SSU96], 1996 [SZ96], 1998 [BBC+98], 2003 [AAB+05], and 2008 [AAB+09]. Continuing this tradition, twenty-eight database researchers and two invited speakers met in October 2013 at the Beckman Center on the University of California-Irvine campus for two days of discussions (see `http://beckman.cs.wisc.edu`). The meeting attendees represented a broad cross-section of interests, affiliations, seniority, and geography. Meeting attendance was capped at thirty so that the meeting would be as interactive as possible.

This year, meeting participants quickly identified Big Data as a defining challenge of our time. Big Data emerged due to the confluence of three major trends. First, it has become much cheaper to generate a wide variety of data, due to inexpensive storage, sensors, smart devices, social software, multiplayer games, and the emerging Internet of Things, which connects homes, cars, appliances, and other devices. Second, it has become much cheaper to process large amounts of data, due to advances in multicore CPUs, solid state storage, inexpensive cloud computing, and open source software. Finally, in a trend called the democratization of data, not just database administrators and developers, but many more types of people have become intimately involved in the process of generating, processing, and consuming data – decision makers, domain scientists, application users, journalists, crowd workers, and everyday consumers.

As a result of these accelerating trends, there is now a widespread realization that an unprecedented volume of data can be captured, stored, and processed, and that the knowledge gleaned from such data can benefit everyone: businesses, governments, academic disciplines, engineering, communities, and individuals. In a sense, the rest of the world has now caught on to the importance of what the database community has been advocating and doing for years.

The new era of Big Data has drawn many communities into the "data management game." There has been a groundswell of efforts in these communities to develop custom data management solutions, such as Hadoop and NoSQL. Many of these early solutions were not based on database management system (DBMS) principles. However, as these solutions have gained popularity and been applied to more data management scenarios, DBMS principles have been increasingly recognized as important and incorporated into solutions. For example, Hive, which manages data declaratively, has become far more popular than MapReduce; new drop-in alternatives to Hive look like parallel DBMSs; NoSQL tools are now moving to high-level languages and ACID transactions; and a new generation of systems is emerging that look like massive relational database management systems running on top of key-value stores that span data centers (e.g., the F-

1 system that powers Google's ad infrastructure).

Thus, today the database community is entering a time of unprecedented excitement. We are now squarely at the center of the Big Data revolution. The world has adopted a vision of a data-driven society, and other communities are adopting DBMS principles. As the community that has been pushing the limits of processing big data sets for 45 years, we can build on a wealth of results, lessons, and experience to help the data-driven world move forward. Our community therefore is uniquely positioned to address Big Data. The opportunity for us to make transformative impact is enormous.

But we also face enormous challenges. Big Data requirements will cause massive disruptions to the ways that we design, build, and deploy data management solutions. The main characteristics of Big Data are volume, velocity, and variety. Our community has worked on volume and velocity for decades, and has developed solutions that are mission-critical to virtually every commercial enterprise on the planet. Big Data, however, brings unprecedented scale that will force us to radically rethink existing solutions. Variety means integrating and analyzing data that come from diverse sources, with varying formats and quality, a topic that we have also been working on for years. However, it is still an extremely labor-intensive journey from raw data to actionable knowledge. This problem will be exacerbated by Big Data, causing a major bottleneck in the data processing pipeline. Hence, we need to intensify our effort to develop end-to-end solutions that scale, are easy to use, and minimize human effort. Big Data also brings wide variety in hardware infrastructures; processing frameworks, languages, and systems; programming abstractions; degrees of user sophistication; and user preferences. Designing data management solutions that can cope with such extreme variety will be a difficult challenge.

Moving beyond the "three V's," many Big Data applications will be deployed in the cloud, both public and private, on a massive scale. Many applications will involve people, e.g., to help solve semantic problems that still bedevil current automatic solutions. The scale of human involvement can range from a single domain expert to a crowd of workers, a whole user community, or in some cases the entire connected world (e.g., Wikipedia). These new trends raise novel and important research challenges for our community.

Finally, Big Data brings important community challenges. We must rethink our approach to teaching data management technologies, reexamine our research culture, and consider the emergence of data science as a discipline. Other aspects of the Big Data revolution, such as the impact on privacy, new ideas about data valuation and ownership, and the emerging data economy, also need to be considered and may affect our training programs and research agendas. However, we will not focus on these aspects in this report.

Over the two days of the Beckman meeting, participants extensively discussed the above issues. Sections 2 and 3 summarize the discussions about research and community challenges, respectively. Section 4 concludes the report.

2. RESEARCH CHALLENGES

The meeting identified five Big Data challenges: scalable big/fast data infrastructures; coping with diversity in the data management landscape; end-to-end processing and understanding of data; cloud services; and the roles of people in the data life cycle. The first three challenges deal with the volume, velocity, and variety aspects of Big Data, while the remaining two deal with deploying Big Data applications in the cloud and managing the involvement of people in these applications.

These Big Data challenges are not an exclusive agenda to be pursued at the expense of other existing work. In recent years our community has strengthened core competencies in RDBMSs, and branched out into many new data management directions, in collaboration with other communities (e.g., systems, AI, KDD, HCI, and e-science). These thriving directions require continued investigation. In addition, important issues that were raised repeatedly during the meeting include security, privacy, data usage and pricing, data attribution, social and mobile data, spatio-temporal data, personalization and contextualization, energy constraints, and scientific data management. Many of these issues cut across the identified challenges and are captured in various aspects of the discussion below.

2.1 Scalable Big/Fast Data Infrastructures

Our community has long been developing systems for processing data in volumes that push the limits of current hardware. Hardware continues to evolve, bringing new processor, storage, and networking technologies. We must continue to address the challenge of building scalable systems to manage bigger data sets that arrive at increasing speed, leveraging these new and improved technologies.

In the database world, the parallel processing of large structured data sets has been a major success, leading to several generations of commercial SQL-based products that are widely used by enterprises.

The distributed computing field has achieved success in scaling up data processing for less structured data on large numbers of unreliable, commodity machines through the use of constrained programming models such as MapReduce. Higher level languages, inspired by declarative database languages such as the relational algebra and SQL, have followed. These have been layered on top of the earlier constrained models, to enable a broader audience of developers to use scalable Big Data platforms. Today, open source platforms such as Hadoop – with its MapReduce programming model, large-scale distributed file system (HDFS), and higher level languages (e.g., Pig and Hive) – are seeing rapid adoption for processing less structured data, even in the traditional enterprise world.

Given the enthusiastic adoption of declarative languages for processing Big Data, there is a growing recognition that more general, database-style query processing techniques are needed. These include cost-aware query optimizers and set-oriented query execution engines. Processing much higher data volumes with acceptable response times will require very high degrees of parallelism. Effective query processing strategies will need to fully exploit large clusters of many-core processors, scaling both "up" and "out" in order to meet the anticipated needs. This will create challenges not only for query optimization and execution, but also for progress monitoring, so that a user can diagnose and manage queries that are running too slowly or consuming excessive resources. To adapt to the characteristics of previously unseen data, as well as to reduce the cost of data movement between stages of data analysis, query processors will need to integrate data sampling, data mining, and machine learning computations into their flows.

At data center scale, the ratio between the speed of sequential processing and interconnects is changing with the advent of faster networks, full bisection bandwidth networks between servers, and remote direct memory access (DMA) capabilities. In addition to clusters of general-purpose multicore processors, more specialized processors should be considered. Commercially successful database machines have demonstrated the potential of hardware-software co-design for data management. Researchers should continue to explore ways of leveraging specialized processors, e.g., graphics processing units (GPUs), field-programmable gate arrays (FPGAs), and application specific integrated circuits (ASICs), for processing very large data sets. These changes in communications and processing technologies will require a reconsideration of parallel and distributed query processing algorithms, which have traditionally focused on more homogeneous hardware environments.

Turning to storage, the database research community must learn how best to leverage emerging memory and storage technologies. Relative to commodity magnetic disks, solid-state disks are expensive per gigabyte but cheap per I/O operation. Various non-volatile random-access memory (NV-RAM) technologies are under development, all with different speed, power, and durability characteristics. Both server-attached and network-attached storage architectures need to be considered. Distributed file systems like HDFS, which are server-attached yet shared across the network, are a hybrid of both approaches. How best to use this range of storage configurations reopens many questions reminiscent of past debates of shared memory vs. shared disk vs. shared nothing, questions that many have considered to be "closed" for traditional parallel relational systems.

To process data that arrives at ever higher speeds, new scalable techniques for ingesting and processing streams of data will be needed. Algorithms will need to be tuned carefully according to the behavior of hardware, e.g., to cope with non-uniform memory access (NUMA) and limited transfer rates across layers of the memory hierarchy. In addition, the very high speed of some data sources, often with lower information density, will require some data to be processed online and then discarded without being persisted in its entirety. Rather, samples and aggregations of such data will need to be selected to be stored persistently in order to answer certain categories of queries that arrive after the raw data is no longer available. For such data, progressive query processing will be increasingly important to provide incremental and partial results with increasing accuracy as data flows through the processing pipeline.

For data that is persisted but processed just once (if ever), it makes little sense to store and index the data first in a database system. For such data, schema-on-read may make more sense than traditional schema-on-write, which imposes unnecessary overhead at ingestion time. At that time one may just want to dump the bits without hassle, returning if and when one wants to interpret the bits. Further, the appropriate way of interpreting the data for a given query may depend on the query, and hence may be unknown at write time. Raw files (i.e., array-of-characters or array-of-bytes) are the least common denominator for interoperation among the wide variety of systems being brought to bear on

data today. As a result, we need to develop tools and languages to assist with schema-on-read, and query engines that run efficiently over raw files.

In addition to much broader data analysis requirements, today's world brings new requirements for data capture, updates, and fast (but simple) data access. Handling high rates of data capture and updates for schema-less data has led to the development of NoSQL systems. The current Big Data platform landscape contains a number of such systems, with nearly as many transaction models and data consistency guarantees as there are examples of such systems. Most provide only basic data access capabilities and weak atomicity and isolation guarantees, making it difficult to build and reason about reliable applications. As a result, a new class of Big Data system has emerged, one that provides full-fledged database-like features over key-value stores or similar substrates. For some applications, the stored data is still managed and updated as "the source of truth" for an enterprise. In other cases, such as the Internet of Things, the stored data reflects and needs to keep up with events and changes occurring in the outside world so that applications can respond to important events or recognize situations of interest. This creates an opportunity for our community to revisit its thinking about data currency and consistency and to design new models and techniques for developing robust applications.

Finally, scalability should be measured not only in terms of petabytes of data and queries per second, but also total cost of ownership (including management, energy use, etc.), end-to-end processing speed (i.e., time from raw data arrival to eventual insights), brittleness (for example, the ability to continue despite failures such as partial data parse errors), and usability (especially for entry-level users). To measure progress against such broader metrics, new types of benchmarks will be required.

2.2 Diversity in the Data Management Landscape

In addition to high data volumes and data arrival rates, today's data-driven world involves a much wider and much richer variety of data types, shapes, and sizes than traditional enterprise data.

In the enterprise world, data has traditionally been stored and analyzed in a data warehouse that has been carefully designed and optimized for repetitive and ad-hoc analysis tasks. In today's more open world, data is often stored in different representations managed by different software systems with different APIs, query processors, and analysis tools. It seems unlikely that a single, one-size-fits-all, Big Data system will suffice for this degree of diversity. Instead, multiple classes of systems will likely emerge, with each addressing a particular class of need (e.g., data deduplication, analysis of large graphs, diverse scientific experiments, real-time stream processing) or exploiting a particular type of hardware platform (e.g., clusters of inexpensive machines, large multicore servers). For these scenarios, database researchers should apply our expertise in set-oriented parallel processing and in efficiently handling data sets that do not fit in main memory.

It remains to be seen how many different types of systems may be needed – e.g., what an appropriate system's scope may turn out to be – but the need for coexistence of multiple Big Data systems and analysis platforms is certain. Thus, another diversity challenge is helping data analysts combine and analyze data across systems. To support Big Data queries that span systems, platforms will need to be integrated and federated. This will involve not only hiding the heterogeneity of data formats and access languages, but also optimizing the performance of accesses that span diverse Big Data systems and of flows that move data between them. We also face a challenge of managing Big Data systems that run on a multitude of diverse devices and reside within or span large data centers. Disconnected devices will also become increasingly common, raising challenges related to reliable data ingestion, query processing over these devices, and data inconsistency in such sometimes-connected, wide-area environments.

Moving up a level, in a diverse and data-driven world, we must manage diverse programming abstractions against very large data sets. Rather than expecting to develop "the" data analysis language for Big Data, perhaps by extending SQL or another popular language, we must let users analyze their data in the medium they find most natural. For example, this may be SQL, Pig, R, Python, a domain-specific language, or a lower-level constrained programming model such as MapReduce or Valiant's bulk synchronous processing model. This requires developing reusable middle-layer patterns, such as scalable matrix multiplication, list comprehension, or iterative execution paradigms, with support for multiple language-specific bindings or embeddings. Another potentially fruitful focus is tools for the rapid development of new domain-specific data analysis languages – tools that simplify the implementation of new scalable, data-parallel languages.

To handle data diversity, then, we need modular platforms that can span both "raw" and "cooked" data, systems where the cooked data can take many forms, e.g., tables, matrices, or graphs. Such systems will run end-to-end dataflows and workflows that mix multiple types of data processing, e.g., querying data with SQL and then analyzing it with R. To help unify systems that access data in such diverse ways, lazy computation is sometimes beneficial – including lazy data parsing/conversion/loading, lazy indexing/view construction, or just-in-time query planning. Big Data systems should become more interoperable and interconnectable, like "Lego bricks." Frameworks like Mesos and now YARN provide inspiration at the systems level, as do workflow systems for the Hadoop ecosystem and tools for managing scientific workflows.

2.3 End-to-End Processing and Understanding of Data

To meet the needs of a data-driven world, the database research community needs to focus on end-to-end processing and understanding of data. Despite years of R&D, surprisingly few tools can process data end-to-end, going from raw data all the way to extracted knowledge, without significant human intervention at each step. Moreover, for most steps, the intervening people need to be highly computer savvy. Few tools in this area are open source. Most are expensive proprietary products that address certain processing steps. As a result, existing tools cannot easily benefit from ongoing contributions by the data integration research community. To overcome this situation, we recommend a focus not just on improving data integration technologies (such as data cleaning, schema matching, and data deduplication), but also on fusing these puzzle pieces into end-to-end solutions.

What should such end-to-end tools look like? At its core, the raw-data-to-knowledge "pipeline" will look much like it always has. Its major steps will continue to be: data acquisition; selection, assessment, cleaning, and transformation (also called "data wrangling"); extraction and integration; mining, OLAP, and analytics; and result summarization, provenance, and explanation. What has significantly changed is the much greater diversity of data and users, and much greater scale. Data today comes in a wide variety of formats. Combinations of structured and unstructured data are appearing that users want to use together in a structured fashion. Further, a wide range of people from many different domains are now building data tools that exploit human feedback in almost every step of the analytical pipeline. Data tools are increasingly used directly by subject-matter experts, not just by IT experts. For example, a journalist with a CSV file of crime statistics may want to clean, map, and publish his or her data. An entirely new class of people devoted to data analysis, called data scientists, has emerged. Once an analytic result has been produced, it is likely to be consumed by a much wider variety of people than before. Finally, data tools are now being used at every imaginable scale, from extracting and combining data from just a few Web pages to mining petabytes of system, network, and application logs and event streams.

Our community should seek to build effective, useful, and impactful tools that can work together, end-to-end. There will likely be no one-size-fits-all tool for the wide variety of data analysis scenarios ahead. We should thus develop multiple tools, each solving some piece of the raw-data-to-knowledge puzzle, which can be seamlessly integrated and be easy to use for both lay and expert users. When possible, we should aim to open source data analysis "building blocks," to be combined and reused by others, and provide best practice guidance on when to use each tool. Tools should handle the range from a small amount of data up to very large volumes. In an increasingly collaborative world for data sharing and analysis, each step of the data analysis pipeline should be interactive and be able to exploit feedback from individuals, teams, and even crowdsourcing.

Tools should be able to exploit domain knowledge, such as dictionaries, knowledge bases, and rules, and be easy to customize to a (new) domain. With a large volume of data to analyze, tool designers should consider using machine learning to partially automate the customization process. Handcrafted rules will remain important, though, as many analysis applications require very high precision, such as e-commerce. In such applications, analysts often write a large number of rules to cover "corner cases" that are not amenable to learning and generalization. To be truly end-to-end and easy to use, tools should provide support for writing, evaluating, applying, and managing hand-crafted rules.

Explanation, provenance, filtering, summarization, and visualization requirements crop up in all steps of the raw-data-to-knowledge pipeline. They will be critical to making analytic tools easy to use. Capturing appropriate meta-information is key to enable explanation, provenance, and reuse. Furthermore, visualization provides an essential way to interact with and solicit input from people, and it can be especially effective when coupled with automatic analysis techniques. Visual analytics is re-

ceiving growing attention in the database, HCI, and visualization communities, for visualizing database queries, visual data mining, and data wrangling. This area would benefit from attention, as it is a must for coping with Big Data volumes.

Analytical data management is knowledge-intensive. The more knowledge we have about a target domain, the better that tools can support the domain's analyses. As a result, there has been a growing trend to create, share, and use domain knowledge to better *understand* data. Such knowledge is often captured in knowledge bases (KBs) that describe the most important entities and relationships in a domain. For example, a community of domain scientists, say in biomedicine, may build a large KB that contains profiles of tens of thousands of biomedical researchers along with their publications, affiliations, and patents. Such KBs are used for improving the accuracy of the raw-data-to-knowledge pipeline, answering queries about the domain, and finding domain experts. Many companies have also built KBs for answering user queries, annotating text, supporting e-commerce, and analyzing social media.

The KB trend will likely accelerate, leading to a proliferation of "knowledge centers" built, maintained, and used by online communities, companies, and others. Such centers will contain knowledge bases as well as tools to query, share, and use them for data analysis. Many of these tools will be invocable in the cloud, allowing users and applications in the same domain and beyond to use the knowledge centers. End-to-end processing from raw data to knowledge will require our community to pay increased attention to this trend, as it can be viewed as using domain knowledge, often a great deal of it, to better understand the raw data in terms of the entities and relationships in its domain. To date we have made some inroads into this topic (e.g., efforts to build KBs in various domains) and have had some significant success (e.g., YAGO). However, more needs to be done, including developing solutions to let a group of users build and maintain a domain-specific KB, to let raw-data-to-knowledge tools utilize such KBs, and to allow users, from layman to experts, to easily query and share such KBs.

2.4 Cloud Services

Cloud computing has become mainstream. Enterprises have a multitude of cloud providers to choose from. Cloud computing has a wide variety of forms, including IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service). Moreover, the distinctions among IaaS, PaaS, and SaaS have started to blur. For example, IaaS providers nowadays provide manageability features that begin to resemble PaaS. From a data platform perspective, the ideal goal is to provide PaaS in its truest form. In a world with true PaaS for data, users would be able to upload data to the cloud, query it exactly as they do today over their SQL databases on the intranet, and selectively share the data and results easily, all without worrying about how many instances to rent, what operating system to run on, how to partition the databases across servers, or how to tune them. Despite the emergence of services such as Database.com, Google Big Query, Amazon Redshift, and Microsoft Azure SQL Database, we are still far away from that vision. Below we outline some of the critical challenges for our community in realizing the vision of Data Platform as a Service in the cloud.

The first challenge is elasticity. While computation is elastic in many cases, data is not. In today's architectures, data can be prohibitively expensive to move. Given this reality, if we want to build an elastic Data Platform as a Service, how should it be architected, keeping in mind the evolution of storage and networking? Should storage be server-local or network-attached? Can the same cloud storage service support both transactions and analytics? How does caching fit into the picture? Handling elasticity also requires leveraging the availability of additional resources as well as preemption of existing resources. Database engines and analysis platforms for a Data Platform as a Service will need to operate on top of elastic resources that can be allocated quickly during workload peaks but possibly pre-empted for users paying for premium service.

Data replication is another challenge. Although data replication has been studied extensively in the past, it is important to revisit it in the context of the cloud, keeping in mind the need for high availability, load balancing, and cost. Both elasticity and replication need to be considered not just within, but also across, geographically distributed data centers.

System administration and tuning is a third challenge. A data platform in use as a cloud service will need extreme auto-tuning. In the world of Data Platform as a Service, the traditional roles of database and system administrators simply do not exist. Therefore, all administrative tasks such as capacity planning, resource provisioning, physical data management, and admission control policy setting need to be automated while dealing with the variance that arises due to the elasticity of resources and their availability in the cloud setting.

Multitenancy is a key technical challenge in man-

aging elasticity for data-related services. To be competitive, the provider of a Data Platform as a Service must offer a cost structure comparable to or better than an on-premises solution. This requires providers to pack multiple tenants of a database service together to share physical resources on the same server to smooth demand and reduce cost. However, multitenancy introduces two problems. First, providers must be able to provide performance isolation so that a burst of demand from one tenant does not unduly degrade the performance of others. This requires careful governance of CPU, I/O, memory, and network resources. Second, users of a database service must be given security guarantees against information leakage across tenants.

Service Level Agreements (SLAs) are critical but challenging in the world of cloud services. For a multitenant Data Platform as a Service, the elasticity of global resource availability as well as the need for resource governance impact the availability of resources for a tenant. In turn, such variations can affect the quality of service. We are just beginning to understand the interaction between multitenant resource allocation and quality of service. Such an understanding would help form the basis for differentiated SLAs for Data Platform as a Service. Today, SLAs primarily focus on availability. To make Data Platform as a Service ubiquitous, it is important to understand this key question deeply, as it has implications not only for cost structures for tenants, but also for the development of QoS-aware database applications based on cloud services. In addition, cost structures for differentiated services must be easy to comprehend in user terms.

Data sharing is another key challenge, as the cloud enables it at an unprecedented scale. The database community should seek to develop novel services that harness this potential. We have already seen services that enable collaborative productivity tools as well as the ability to share results of data analysis or visualization. There is a great opportunity for us to actively explore richer ideas in the context of data analytics. For example, what would collaborative data analytics look like in the future? To realize such a vision, we must understand how we can support essential services such as data curation and provenance when we want to perform such activities collaboratively in the cloud. Data sharing in the cloud will also raise new issues in leveraging data sets, such as how to find useful public data, how to correlate your own data with public data to add context, how to find high-quality data in the cloud, and how to share data at fine-grained levels, as well as business issues, such as how to distribute costs when sharing computing and data and how to price data. The cloud will create new life-cycle challenges, such as how to protect data if the current cloud provider fails, or how to preserve data for the long term when it lives "somewhere out there." The cloud will also drive innovation in tools for data governance, such as auditing, enforcement of legal terms and conditions, and explanation of user policies.

Hybrid clouds bring a new set of challenges as well. Today, this entails support for sharing and seamless operation between database services and servers that reside on-premise and those in a single cloud provider. In the future, data sharing services will need to be federated across mobile devices, on-premise resources, and multiple cloud providers. We also need to support common patterns of hybrid clouds, e.g., organizations may run applications in their private cloud during normal operation, but then tap into a public cloud at peak times or when unanticipated events bring surges in load. Another example is cyber-physical systems, as in the Internet of Things, where, e.g., cars will upload data into a cloud and obtain control information in return. Cyber-physical systems involve data streaming from multiple sensors and mobile devices, and must cope with intermittent connectivity and limited battery life, which pose difficult challenges for real-time and perhaps mission-critical data management in the cloud.

2.5 Roles of Humans in the Data Life Cycle

Back when data management was an enterprise-driven activity, it was clear who did what: developers built databases and database-centric applications, business analysts queried databases using (SQL-based) reporting tools, end users generated data and queried and updated databases, and database administrators tuned and monitored databases and their workloads. Today, the world has dramatically changed. A single individual may now play multiple roles in the data life cycle, and many Big Data applications involve people in many different roles. The database research community must address this change, managing not just the data, but the people as well.

There has been a growing recognition of the increasing role of people in the data life cycle, of course, such as the work done in our community and elsewhere on crowdsourcing. However, the new need to "manage the people" is not just about crowdsourcing or micro-tasks (i.e., tasks that take a crowd worker a few minutes to perform). Today's land-

scape requires the consideration of people (and human factors) as they relate to query understanding and refinement, identifying relevant and trustworthy information sources, defining and incrementally refining the data processing pipeline, and visualizing relevant patterns and obtaining query answers, all in addition to making the various micro-tasks doable by domain experts and end users. We can classify people's roles into four general categories: producers of data, curators of data, consumers of data, and community members. Below we discuss each category and its associated data management research challenges.

Many people today are data producers, as virtually anyone can generate a torrent of data now through the sharing of tables, the use of mobile phones, social platforms and applications (e.g., Facebook, Twitter), and an increasing collection of wearable devices (e.g., Fitbit). One key challenge for the database community is to develop algorithms and incentives that guide people to produce and share the most useful data, while maintaining the desired level of data privacy. For instance, when people produce data, how can we help them add metadata quickly and accurately? As one example, when a user uploads an image, Facebook automatically identifies faces in the image so that users can optionally tag them. As another example, there are tools to automatically suggest tags for a tweet. What else can we do, and what general principles and tools can we provide?

More people are becoming data curators. In today's data-driven world there is less central control over data. Data is no longer just in databases controlled by a DBA and curated by the IT department. Instead, as mentioned earlier, a wide variety of data is now being generated, and a wide variety of people are now empowered to curate it. In particular, crowdsourcing has emerged as a promising curation solution. Another key challenge, then, is to obtain high-quality data sets from a process based on often-imperfect human curators. Two related challenges are building platforms that allow people to curate data easily and extending relevant applications to incorporate such curation. For these people-centric challenges, data provenance and explanation will be crucial, as will considerations of privacy and security.

People are data consumers as well. Increasingly, people want to use messier and messier data in complex ways. This raises many challenges. In the enterprise, data consumers have usually been people who know how to ask SQL queries, via a command-line interface or a graphical query tool, over a structured database. Today's data consumers may not know how to formulate a query at all – e.g., a journalist who wants to "find the average temperature of all cities with population exceeding 100,000 in Florida" over a structured data set. Our community's challenge is to make it possible for such people to get their answers themselves, directly. This requires new query interfaces, e.g., interfaces based on multitouch, not just console-based SQL interfaces. We need interfaces that combine visualization, querying, and navigation. Many data consumers may not know what queries to ask, and the available data may or may not support their needs. When the query to ask is not clear, people need other ways to browse, explore, visualize, and mine the data. We must build tools and infrastructures that make the data consumption process easier, including the notions of trust, provenance, and explanation, and we must target the diverse user base of the emerging data-driven world.

People are community members. Numerous communities exist online, with more being created daily. Members of such communities often want to create, share, and manage data, and it is becoming increasingly easy for them to do so. In particular, members may want to collaboratively build community-specific knowledge bases, wikis, and tools to process data. For example, many researchers have created their own pages on Google Scholar, thereby contributing to this "community" knowledge base. Our challenge is to build tools to help communities produce usable data as well as to exploit, share, and mine it.

3. COMMUNITY CHALLENGES

In addition to research challenges, the meeting also discussed a host of community issues. These include database education, research culture, and data science and scientists. Some of these issues are new, brought about by Big Data. Other issues, while not new, are exacerbated by Big Data and are becoming increasingly important for our community to address.

One issue that meeting participants discussed was database education. The way we teach database technology today is increasingly disconnected from reality. We still teach the technology of the 1980's, when memory was small relative to data sizes, making I/O a costly portion of database operation, and when computation was also quite expensive. Today, however, the world looks very different. Technological advances have turned many previous design constraints upside-down. For example, databases can now be entirely memory resident for some classes

of applications; new storage technologies eliminate some of the sequential vs. random I/O issues of the past; and advances in distributed computing have brought us self-managing distributed file systems, scalable parallel data processing techniques, and new declarative languages. While influenced by database languages, these new languages relax some of SQL's rigidity and are compiled down to MapReduce jobs on existing execution platforms.

Despite these changes, we still base our teaching on the architectural blueprint born in the 1970's and 1980's. Similarly, our data model and query language teachings focus on relations and SQL. There was a widely shared sense at the meeting that change in database education is overdue, but no consensus on what this change should be. Some suggested that we start with new technologies, e.g., columnar instead of row-based storage, while others felt that we should move to teaching top-down and explore the alternate technologies available at each architectural decision point, or to start with notions of data quality and value in the bigger picture of going from raw data to knowledge. In addition, functionality once limited to databases and hidden under SQL is now appearing in different contexts and becoming available in smaller, more specialized systems (e.g., key-value stores, stream databases), as well as outside of databases (e.g., the use of hash-based parallelism and scalable external sorting in systems like Hadoop). As a result, there was a feeling that we should be teaching the principles, patterns, and algorithms that have come from years of database research as things whose modern applicability is much broader than just SQL system internals. Other questions raised include: how do we "parcel out" the nice "nuggets" buried in the relational tradition so they are not continually reinvented? What about the role of this material in a computer science education – should it be "ghettoized" in a database class, or be pushed into introductory curricula alongside recursion, divide-and-conquer, and object-oriented programming? If we achieve this, what other material should we substitute in the database class?

Besides database education, there is also a concern regarding our research culture. In recent years there has been an alarming increase in emphasis on publication and citation counts instead of research impact. This discourages large systems projects, end-to-end tool building, and sharing of large data sets due to the longer times required and the resulting lower publication density. Program committees (PCs) often value novelty over utility or potential impact. (Publication pressure has also led to huge PCs with no face-to-face meetings, reducing individual accountability and making it difficult for junior PC members to learn from more senior ones.) These problems jeopardize our Big Data agenda. To pursue this agenda effectively, it is important that we develop and share large systems, end-to-end tools, and data sets, to help evaluate and drive our research, and to have practical impact. The field should strive to return to a state where fewer publications per researcher per time unit is the norm, and where large systems projects, end-to-end tool sets, and data sharing are more highly valued. However, there was no consensus on how best to get there from here – something to grapple with over the incoming years.

Another major change is that Big Data has generated a rapidly growing demand for data scientists: individuals with skills to transform large volumes of data into actionable knowledge. Data scientists need skills not just in data management and large-scale data processing tools and platforms, but also in business intelligence, computer systems, mathematics, statistics, machine learning, and optimization. They also need an ability to work closely with domain experts. In response to this need, some universities are creating data science institutes and degree programs to foster collaboration and assemble the requisite interdisciplinary knowledge and course offerings. The database research community has much to offer such efforts, and we should actively do so. Data science is a cross-disciplinary movement, so participation will require collaborations with domain specialists. Big Data presents computer science with an opportunity to influence the curricula of chemistry, earth sciences, sociology, physics, biology, and many other fields. The small computer science parts in those curricula could be grown and re-adjusted in their focus to give data management and data science a more prominent role.

4. GOING FORWARD

This is an extremely exciting time for database research. In the past we have been guided by, but also restricted by, the rigors of the enterprise, its relational data, and our relational database system architectures. The rise of Big Data and the vision of a data-driven world raise many exciting opportunities and pose many new challenges for the database research community. Being the community that has traditionally dealt with all things related to data, there is a golden opportunity for us to play a central role in this emerging world. There is an abundance of research opportunities related to handling the many challenges of Big Data; of data diversity; of

new hardware, software, and cloud-based platforms; of addressing the data life cycle, from the creation of data to analysis and sharing; and of facing the diversity, roles, and number of people related to all aspects of data. It is also time to rethink our approach to education, our degree of involvement with the consumers of our work, and our value system and its impact on what (and how) we disseminate and how we fund our research.

Acknowledgments: The Beckman meeting was supported financially by donations from the Professor Ram Kumar Memorial Foundation, Microsoft Corporation, and @WalmartLabs.

5. REFERENCES

[BDD+89] Philip A. Bernstein, Umeshwar Dayal, David J. DeWitt, Dieter Gawlick, Jim Gray, Matthias Jarke, Bruce G. Lindsay, Peter C. Lockemann, David Maier, Erich J. Neuhold, Andreas Reuter, Lawrence A. Rowe, Hans-Jorg Schek, Joachim W. Schmidt, Michael Schrefl, Michael Stonebraker. "Future Directions in DBMS Research - The Laguna Beach Participants." *ACM SIGMOD Record*, 18(1):17-26, 1989.

[SSU91] Avi Silberschatz, Michael Stonebraker, Jeff Ullman. "Database Systems: Achievements and Opportunities." *Communications of the ACM*, 34(10):110-120, 1991.

[SSU96] Avi Silberschatz, Mike Stonebraker, Jeff Ullman. "Database Research: Achievements and Opportunities into the 21st Century." *ACM SIGMOD Record*, 25(1):52-63, 1996.

[SZ96] Avi Silberschatz, Stan Zdonik, et al. "Strategic Directions in Database Systems – Breaking Out of the Box." *ACM Computing Surveys*, 28(4):764-778, 1996.

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

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

[AAB+09] Rakesh Agrawal, Anastasia Ailamaki, Philip A. Bernstein, Eric A. Brewer, Michael J. Carey, Surajit Chaudhuri, AnHai Doan, Daniela Florescu, Michael J. Franklin, Hector Garcia-Molina, Johannes Gehrke, Le Gruenwald, Laura M. Haas, Alon Y. Halevy, Joseph M. Hellerstein, Yannis E. Ioannidis, Hank F. Korth, Donald Kossmann, Samuel Madden, Roger Magoulas, Beng Chin Ooi, Tim O'Reilly, Raghu Ramakrishnan, Sunita Sarawagi, Michael Stonebraker, Alexander S. Szalay, and Gerhard Weikum. "The Claremont Report on Database Research." *Communications of the ACM*, 52(6):56-65, 2009.

A Panorama of Imminent Doctoral Research in Data Mining

Aparna S. Varde
Dept. of Computer Science
Montclair State University
Montclair, NJ, USA
vardea@montclair.edu

Nikolaj Tatti
HIIT, Dept. of Information and Computer Science
Aalto University, Finland
nikolaj.tatti@aalto.fi

ABSTRACT

As databases head towards data streams, discovering knowledge from the data poses challenges. The need to process and mine data is affected by the big data wave, advances in Web technology and other factors. The advent of the cloud with related technologies provides further momentum to enhance knowledge discovery. Data mining is also of interest to research communities outside computer science as there is a need to harvest data from various domains incorporating domain-specific factors. These and other issues motivate PhD students to pursue core and applied research in data mining along with stream data management, big data, cloud computing, Web knowledge discovery, domain-specific techniques and more. A PhD forum on data mining provides an excellent platform for doctoral students to present imminent research and get valuable feedback from experts. In addition it gives them the opportunity to disseminate the results of their work among fellow researchers, and publish their novel contributions at an early stage. IEEE ICDM hosts such a PhD forum for doctoral students with a data mining focus. This article describes the content of the work presented at the ICDM 2013 PhD forum. It also gives a brief overview of the organization of this forum. The article thus provides a panorama of recent doctoral student work in data mining that would be of interest to researchers in database management and related areas.

1. INTRODUCTION

The IEEE International Conference on Data Mining (ICDM) hosted its third PhD forum on December 7, 2013. This was after two PhD forums in 2011 [http://webdocs.cs.ualberta.ca/~icdm2011/phd-forum.php], and again the following year in 2012 [http://icdm2012.ua.ac.be/content/phd-forum]. The ICDM 2013 PhD forum [1] was held in conjunction with the main ICDM conference at Dallas, Texas from December 7 to 10, 2013. The PhD forum was co-chaired by Dr. Aristides Gionis, Associate Professor in the Department of Information and Computer Science at Aalto University, Finland, and Dr. Aparna Varde, Associate Professor in the Department of Computer Science at Montclair State University, USA. Dr. Nikolaj Tatti, a Post-Doctoral Researcher in Computer Science at Aalto University, Finland, served as a session chair in the forum.

The keynote talk was given by Dr. Jilles Vreeken from Max Planck Institute for Informatics, Saarbrucken, Germany. Dr. Vreeken is a Senior Researcher in their Databases and Information Systems Group and an Independent Research Group Leader of their Exploratory Data Analysis Group. His talk titled "Don't Panic: The Grad Student's guide to a PhD in Data Mining" [1] served as a motivating and humorous account of the road ahead for an early PhD student. He provided statistical data on the usefulness of a PhD, compared it with other advanced degrees and emphasized the milestones and challenges along the PhD path, with helpful career advice. He also stressed the importance of choosing the right advisor, research areas and core dissertation focus. This involved keeping in mind the students' passion for research as well as the job market in order to tap the practical angle.

This PhD forum attracted around 25 research paper submissions from various parts of the world. Among these, five submissions were selected as full papers and four as short papers. All papers were provided with oral and poster presentations, the full papers with a time slot of 20 minutes and short papers with 10 minutes followed by an interactive question-answer session for each paper. During this, useful feedback was offered by the audience comprising established research professionals and other PhD students. The PC members comprised a team of experts in data mining from academia and industry. There were 19 PC members from 10 countries across the globe.

A highlight of this forum was that despite unexpected weather conditions termed as a statistical outlier in Dallas during December, 100% of the speakers

attended and presented their work. This was a sheer example of the inspiration that the PhD forum provided to the doctoral students at the IEEE conference ICDM.

The work of the students presented at this forum spanned various areas including stream data mining, knowledge discovery from big data and domain-specific issues. The papers focused on topics such as MapReduce for classification, anomaly detection, data stream clustering, mining health records, financial news quantification, local distance metrics, discrete pattern mining, dynamically evolving concepts and time-sensitive route-planning.

The themes of stream data mining and domain-specific data mining seemed to be prevalent among many of the presentations. This indicated the enthusiasm among current PhD students to explore streaming data in addition to the traditional databases addressing challenges such as infinite length, scalability, feature evolution and concept drift in continuous data streams. The fact that domain-specific data mining formed the theme of several papers indicated that data mining research has spread across multiple disciplines and that applied research in data mining has acquired significance in addition to core research. The theme of big data also seemed to be noticed among some papers. This implied that knowledge discovery from data of the order of terabytes and more in conjunction with classical data mining techniques has gained importance among data mining researchers.

Based on the papers presented at the PhD Forum, the rest of this article is organized as follows. We outline a survey of the papers from the forum in Section 2. This is placed in the categories of core research and applied research in data mining respectively based on the primary contributions of the papers. Section 3 presents a discussion including a list of open issues that provide the scope for further research in data mining and related areas. Section 4 gives the conclusions along with the motivation to organize more such events. The acknowledgments and references appear thereafter.

2. SURVEY OF PAPERS

We divide the papers into two categories, namely, “Core Research” and “Applied Research”, respectively, based on their major impact. The Core Research papers are those that have their primary focus on contributing to data mining techniques. The papers in the Applied Research area are the ones that propose novel adaptations of data mining addressing the challenges therein leading to significant contributions.

2.1 Core Research in Data Mining

Anomaly Detection with Clustering: The issue of detecting anomalies is an important aspect of computer security and was addressed in the paper by Mustafa et al. The authors proposed learning techniques in the area of clustering for host-based anomaly detection [1]. In one technique CMN (clustering with Markov network), they clustered benign or secure data in the training phase and from each cluster, built an individual Markov network for modeling benign behavior. In the testing phase each Markov network found the probability of every testing instance. If this probability as calculated from many Markov networks was low, the concerned point was classified as malicious or insecure. Other techniques proposed were CMN-OS (clustering with Markov networks with outlying subspace) and CLP (Clustered Label Propagation). In their experimental evaluation they proved that these approaches were not very sensitive to noise and outperformed other state-of-the-art methods for anomaly detection.

Multi Density Clustering for Streams: Another paper that dealt with clustering was on MuDi-Stream by Amini et al. [1], a multi density-based clustering algorithm to handle streaming data with noise. Streams are continuous and need to be processed with limited time and memory. Also, data of varying densities needs to be clustered. Both these needs were addressed in the proposed algorithm MuDi-Stream which performed clustering in online and offline phases. The online phase developed core-mini-clusters with a new proposed core distance based on number of data points around the core. The offline phase conducted clustering on the core-mini-clusters with a density-based method. Since the algorithm had different core distances for different clusters, it covered some multi-density environments.

MapReduce for Stream Classification: The work by Haque and Khan [1] further delved into the issue of stream data mining. The infinite length and evolving nature of data streams poses challenges in mining. These challenges were addressed in this paper by a multi-tiered ensemble-based method. Several AdaBoost ensembles were built for each numeric feature upon receiving each chunk of the data stream. This was expected to cause scalability problems for huge data chunks and/or too many numeric attributes. Thus, the authors exploited the parallelism of MapReduce in order to propose two approaches to build ensembles such that they enhanced scalability, yet also maintained accuracy. Experiments on benchmark datasets indicated that these approaches were indeed efficient, scalable and accurate.

Tracking Evolving Data Streams: The mining of data streams is affected by factors such as feature evolution, concept drift and novel class emergence. This requires rapid labeling and tracking of such dynamically evolving streams, which was addressed in the paper by Parker and Khan [1]. Feature evolution consists of features getting added, removed or altered in ranges. Concept drift implies that the concepts defining class labels can change over the span of the data. Previously unknown classes can also appear in the data stream which is called novel class emergence. The authors developed approaches for tracking and labeling these streams while adhering to the required constraints of the continuous data. They developed an adaptive supervised ensemble to predict instance labels and a stream clustering approach to monitor new characteristics defining the concepts and new classes that emerge accordingly. Thus, new classes with unknown labels were not treated as noise but instead appropriately labeled. They tested the accuracy and efficiency of their data stream mining approaches by comparison with baseline methods on benchmark data streams.

Discrete Pattern Mining with Matrices: Jiang and Health [1] considered binary matrix factorization where the goal was to approximate a binary matrix as a product of two binary matrices. They argued that a natural approach to force the product to be binary was to force one of the matrices have only one per column. They claimed that this made the problem equivalent to clustering and adopted the standard Lloyd's algorithm for discovering such matrices. They tested their method on synthetic and image datasets, and also used the approximate matrices for mining patterns.

2.2 Applied Research in Data Mining

Discriminative Metric for Applications: Mu and Ding [1] considered learning a local discriminative distance metric for real-world applications. More specifically, they focused on discovering Mahalanobis distance that simultaneously minimized distances of neighboring points belonging to the same class while maximizing the distances of neighboring points belonging to a different class. Using these local distances, the authors constructed a kNN-style classifier and applied it to crater prediction from images, crime prediction, and accelerometer data.

Financial News Quantification: Minev [1] presented his ongoing work on the topic of quantifying financial news. The author considered announcements from Federal Reserve from which he extracted features using Natural Language Processing techniques. Once these features were extracted, the goal was to predict the movement of S&P500, a major stock index, within a day of the published announcement.

Time-sensitive Route Planning: Hsieh et al. [1] proposed a framework for recommending tourist routes. In their approach they constructed a score of a route by taking into account the popularity of the place, the most popular visit time, and transition times between the locations. Given a query, a starting point, the authors proposed a heuristic algorithm, and conducted experiments with several baseline algorithms. The authors also conducted a user study and assessed the users' satisfaction for the proposed routes.

EHR Mining: Lo et al. [1] studied how measuring adverse drug reactions can be discovered in electronic health records (EHR). A standard way of detecting such drug reactions was through a spontaneous reporting system. While discovering correlations between drug and symptoms was straightforward in this system as the symptom and the drug is reported in the same report, it was less trivial when electronic health records were used since the data was collected over a period of time. The authors designed a method for discovering such correlations and tested their method on a synthetic electronic health records dataset where they obtained effective results.

3. DISCUSSION

The PhD Forum in ICDM 2013 was organized for the third time and was a highly successful event after two similar events in 2011 and 2012 respectively. It involved presentations from PhD students on a range of topics in data mining and related areas. These included core research in areas such as learning techniques, big data mining and knowledge discovery algorithms as well as applied research on various topics such as electronic health records, route planning, financial news and other real world applications.

Some of the problems discussed at the forum presented the scope for further research in areas such as stream data mining and domain-specific problems. A few potential areas for future work were gathered from the presentation of the papers and the question-answer sessions that occurred thereafter. These are listed herewith as follows:

- Scientific data mining taking into account sensitive information with aspects such as security
- Discovery of knowledge from sensitive data by cloud mining approaches

- Enhancement of stream data mining by integration of clustering and classification
- Advances in big data management and mining with respect to streaming data
- Techniques to address privacy issues in mining electronic health records
- Adaptation of route planning and other location-specific approaches to mobile devices
- Multilingual processing in knowledge discovery for applications such as financial news

These and other related topics could lead to more interesting findings in data mining research.

4. CONCLUSIONS

This article provides a panorama of the research by upcoming doctoral students in data mining. It would be of interest to students and professionals in data mining, databases and related areas. More details of this imminent doctoral research can be found in the respective papers in the proceedings of the ICDM 2013 PhD Forum.

We hope that some of the open research issues emerging from the papers herewith lead to further research via the PhD students' dissertation sub-problems. This would also promote more interactions among data mining researchers through future work in the concerned areas.

We aspire that the PhD Forum remains an annual event at ICDM and is even more successful in the forthcoming years. This would further serve as the motivation to continue organizing events of this nature in various existing and upcoming data mining and database conferences.

5. ACKNOWLEDGMENTS

The authors thank Dr. Aristides Gionis for co-chairing the PhD forum and ICDM 2013 organizers Dr. Diane Cook, Dr. Bhavani Thuraisingham and Dr. Xingdong Wu for hosting this event. We also thank Dr. Jilles Vreeken for giving an encouraging keynote talk. We express our gratitude towards all the PC members for reviewing papers. Finally, we thank the doctoral students for presenting their work at the PhD forum and making it an exciting event.

6. REFERENCES

[1] Aristides Gionis, Aparna Varde eds., ICDM PhD Forum, http://icdm2013.rutgers.edu/phd-forum, 2013.

EuroSys 2015

21-24 April 2015, Bordeaux, France
http://eurosys2015.labri.fr/

Call for Papers

The European Conference on Computer Systems (EuroSys) is a premier international forum for presenting computer systems research, broadly construed. EuroSys 2015 seeks papers on all areas of computer systems research, including, but not limited to:

- Cloud computing
- Database systems
- Dependable systems
- Distributed systems
- File and storage systems
- Language support and runtime systems
- Mobile and pervasive systems
- Networked systems
- Operating systems
- Parallelism, concurrency, and multicore systems
- Real-time, embedded, and cyber-physical systems
- Secure systems, privacy and anonymity preserving systems
- Tracing, analysis, and transformation of systems
- Virtualization systems

Papers will be judged on novelty, significance, correctness, and clarity. We encourage papers that bridge research in different communities. We also welcome experience papers that clearly articulate lessons learnt and papers that refute prior published results. Papers will be provisionally accepted and final acceptance is subject to shepherding by a member of the program committee.

Reviewing will be double-blind, meaning the authors' identities will be hidden from the reviewers. EuroSys applies ACM's policies for plagiarism, submission confidentiality, reviewer anonymity, and prior and concurrent paper submission.

Full submission details will be published online on the conference web site (http://eurosys2015.labri.fr/). In addition to paper presentations, EuroSys 2015 will have a poster session. Submissions for posters will open closer to the conference deadline. Accepted papers will automatically qualify for the poster session, and authors will be strongly encouraged to participate. In addition, accepted papers will be made available to the public one week before the conference.

Authors who are unsure whether or not their submissions might meet these guidelines, or with specific questions about the guidelines, are welcome to contact the program committee co-chairs, via eurosys2015-chairs@labri.fr.

Abstract submission October 3, 2014
Full paper submission October 10, 2014

General Chair

Laurent Réveillère, *LaBRI*

Program Chairs

Tim Harris, *Oracle Labs*
Maurice Herlihy, *Brown University*

Program Committee

Gustavo Alonso, *ETH Zürich*
Mona Attariyan, *Google*
Sorav Bansal, *IIT Delhi*
Miguel Castro, *Microsoft Research*
Rong Chen, *Shanghai JiaoTong University*
Aleksandar Dragojevic, *Microsoft Research*
Roxana Geambasu, *Columbia University*
Andreas Haeberlen, *University of Pennsylvania*
Wenjun Hu, *Yale*
Frans Kaashoek, *MIT*
Panos Kalnis, *KAUST*
Rüdiger Kapitza, *Technische Universität Braunschweig*
Anne-Marie Kermarrec, *Inria*
Christoph Kirsch, *Salzburg*
Dejan Kostic, *KTH Royal Institute of Technology*
Christos Kozyrakis, *Stanford*
Julia Lawall, *Inria/LIP6*
Wyatt Lloyd, *Facebook / University of Southern California*
Harsha V. Madhyastha, *UC Riverside*
Derek McAuley, Horizon, *U Nottingham*
Thomas Moscibroda, *Microsoft Research*
Derek G. Murray
Thomas Neumann, *Technische Universität München*
Mathias Payer, *Purdue University*
Fernando Pedone, *University of Lugano*
Simon Peter, *University of Washington*
Don Porter, *Stony Brook University*
Oriana Riva, *Microsoft Research*
Amitabha Roy, *EPFL*
Mark Silberstein, *Technion – Israel Institute of Technology*
Asia Slowinska, *Vrije Universiteit Amsterdam*
Susan Spence, *HP Labs*
Swaminathan Sundararaman, *Fusion IO*
Michael Swift, *UW Madison*
Serdar Tasiran, *Koç University*
Dan Tsafrir, *Technion – Israel Institute of Technology*
Michael Vrable, *Google*
Robert N. M. Watson, *University of Cambridge*

Call for Participation for SoCC 2014 (ACM Symposium on Cloud Computing)

Call for Participation
Fifth ACM Symposium on Cloud Computing (SoCC)
November 3rd-5th 2014, Seattle, Washington, USA
Web Site: https://sites.google.com/site/2014socc/

The ACM Symposium on Cloud Computing 2014 (ACM SoCC 2014) will be the fifth in a series of symposia that brings together researchers, developers, users, and practitioners interested in cloud computing. The scope of SoCC is broad and encompasses diverse systems topics such as software as a service, virtualization, and scalable cloud data services. ACM SoCC is the premier conference on cloud computing; it is the only conference co-sponsored by the ACM Special Interest Groups on Management of Data (SIGMOD) and on Operating Systems (SIGOPS).

SoCC 2014 will feature a high-quality single track technical program as well as two keynote talks by Tom Anderson from University of Washington and Joe Hellerstein from UC Berkeley and Trifacta. The list of accepted papers will soon be available at https://sites.google.com/site/2014socc/home/program. The conference will also include a poster session, an excellent opportunity to learn about works in progress.

Register by October 5, 2014 to get a discounted rate:
https://sites.google.com/site/2014socc/home/registration

SoCC 2014 will be offering travel and registration scholarships for students interested in attending the conference. More details are available at: https://sites.google.com/site/2014socc/home/student-scholarships

We look forward to seeing you at SoCC 2014!

Cosmin Arad (SoCC 2014 Publicity Chair)

On behalf of:

- SoCC General Chairs: Ed Lazowska (University of Washington) & Doug Terry (Microsoft Research)
- SoCC PC Chairs: Remzi H. Arpaci-Dusseau (Wisconsin) & Johannes Gehrke (Microsoft)
- SoCC Program Committee: https://sites.google.com/site/2014socc/home/pc
- SoCC Organizing Committee: https://sites.google.com/site/2014socc/home/organizers